
AFRL-RI-RS-TR-2015-218

# EFFICIENT TRACKING, LOGGING, AND BLOCKING OF ACCESSES TO DIGITAL OBJECTS

UNIVERSITY OF NORTH CAROLINA

*SEPTEMBER 2015*

FINAL TECHNICAL REPORT

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED***

STINFO COPY

**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2015-218 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

**/ S /**
ROBERT KAMINSKI
Work Unit Manager

**/ S /**
WARREN H. DEBANY JR
Technical Advisor, Information
Exploitation and Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

*Form Approved*
**OMB No. 0704-0188**



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| SEP 2015 | FINAL TECHNICAL REPORT | SEP 2012 – MAR 2015 |

**4. TITLE AND SUBTITLE**

EFFICIENT TRACKING, LOGGING, AND BLOCKING OF ACCESSES TO DIGITAL OBJECTS

**5a. CONTRACT NUMBER**
FA8750-12-2-0235

**5b. GRANT NUMBER**
N/A

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Fabian Monrose, Michael Bailey, Charles Schmitt

**5d. PROJECT NUMBER**
DHS2

**5e. TASK NUMBER**
UN

**5f. WORK UNIT NUMBER**
C1

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
University of North Carolina
Office of Contracts and Grants
104 Airport Dr. STE 2200
Chapel Hill NC 27599-5023

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory/RIG
525 Brooks Road
Rome NY 13441-4505

**10. SPONSOR/MONITOR'S ACRONYM(S)**
AFRL/RI

**11. SPONSOR/MONITOR'S REPORT NUMBER**
AFRL-RI-RS-TR-2015-218

**12. DISTRIBUTION AVAILABILITY STATEMENT**
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

In this project, the performer moved the field of digital provenance forward by designing and implementing techniques for following the chain of custody of data in a virtualized environment. Specifically, the goals were to provide an approach for tracking accesses to objects that originate from disk, and capture subsequent accesses to these objects in memory. To that end, one key capability provided is an accurate, and efficient, tracking and reconstruction mechanism for collating and storing events collected at different levels of abstraction. The effort also provided a rich interface for managing and mining the captured information, thereby providing deeper insights into what transpired after a compromise has been detected (e.g., a suspicious transfer of data to an external device or modification of files). Additionally, capabilities to not only record, but to also deny unauthorized accesses or transfer of data from objects within a protected data store (e.g., a disk partition) for which provenance tracking has been enabled.

**15. SUBJECT TERMS**

digital provenance, tracking, monitoring

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | **ROBERT KAMINSKI** |
| U | U | U | UU | 77 | 19b. TELEPHONE NUMBER (*Include area code*) **N/A** |

# Contents

# List of Figures

# 1 Executive Summary

Today, postmortem analysis of security events is an all too familiar problem. Our workstations and other computing devices are constantly at risk from compromise while performing seemingly benign activities like browsing the Web, interacting on social-networking websites, or by abuse from malicious actors that use compromised hosts as platforms for various nefarious activities. Sometimes, the threats can also arise from the inside (e.g., corporate espionage and other insider threats), and can lead to substantial financial losses. Underscoring each of these breaches is the need to reconstruct past events to know *what* happened and to better understand *how* it happened. Sadly, although there has been significant improvements in computer systems over the last few decades, the problem of data provenance, and in particular, data forensics has been largely ignored.

In this project, we moved the field of digital provenance forward by designing and implementing techniques for following the chain of custody of data in a virtualized environment. Specifically, we provided an approach for *tracking* accesses to objects that originate from disk, and capture subsequent accesses to these objects in memory. One key capability we provided is an accurate, and efficient, tracking and reconstruction mechanism for collating and storing events collected at different levels of abstraction. We also provided a rich interface for managing and mining the captured information, thereby providing deeper insights into what transpired after a compromise has been detected (e.g., a suspicious transfer of data to an external device or modification of files). Additionally, we provided capabilities to not only record, but to also deny unauthorized accesses or transfer of data from objects within a protected data store (e.g., a disk partition) for which provenance tracking has been enabled. The violations in access patterns may be expressed in policies that include conditions describing what set of applications are allowed to access the monitored object. A high-level depiction of some of our capabilities is shown in Figure 1.

## Performance Goals

Undoubtedly, digital provenance is a problem of substantial importance to DHS. Under this con- tract, we push the boundaries beyond the current state of the art and provide new techniques that enable DHS and its customers to: (*a*) identify and authenticate access to digital objects (*b*) track accesses to these objects on disk and in memory, and (*c*) track edits to these objects via a provenance-aware audit trail. We also demonstrated the value of these techniques through evaluation within the Secure Medical Workspace (SMW) Project led by our partners at the Renaissance Computing Institute (RENCI). The Secure Medical Workspace provides a secure centralized infrastructure with virtualization and data leakage protection technologies to allow researchers to manipulate and analyze research data while ensuring that sensitive data remains within the SMW environment, and to ensure that any sensitive data is only accessed by pre-approved applications.

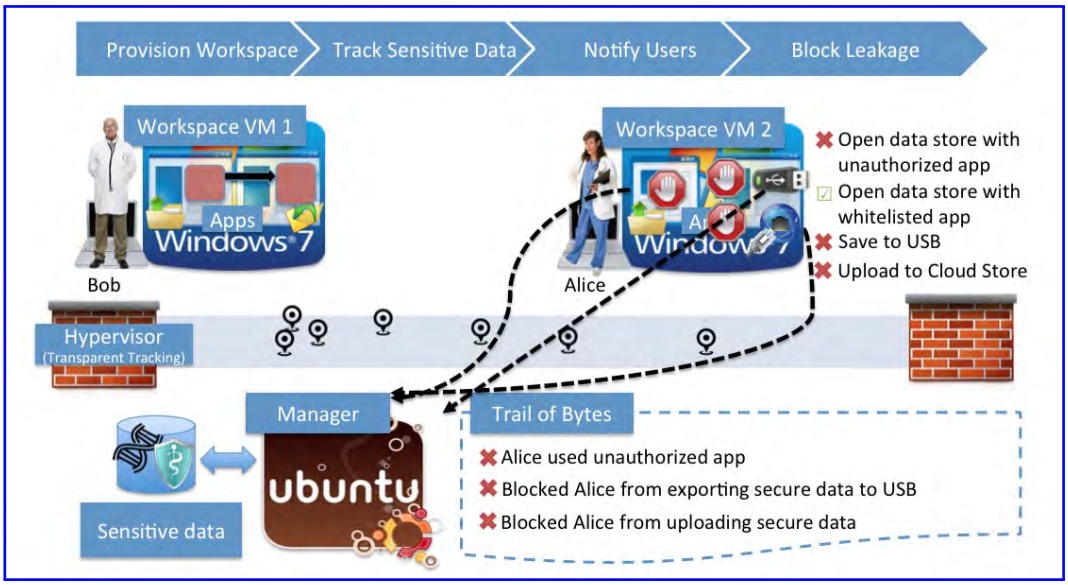


Figure 1: Conceptualization of the final provenance-aware audit system delivered.

# 2 Introduction

The list of technical tasks for this project is presented below. Thereafter, we review the goals that serve as guides for the methods, assumptions and procedures sections that follow.

- Ontological Provenance Model: Provide an ontological model for data provenance representation. These representations inform the management and presentation aspects of this work. Clearly, the problem of data provenance of an item has been studied in a variety of settings, each of which have suggested ways of thinking about this problem—especially as it relates to support for queries on provenance (i.e., collecting provenance on data is not useful unless that provenance is easily accessible). We explore how to best represent and manage provenance information for objects both in memory and on disk.

- Provenance Tracking: Develop heuristics for allowing our tracking framework to coalesce the events collected at various layers of abstraction, and to map these events back to the offending process(es). These mappings, as well as memory snapshots of the processes that accesses the objects of interest will recorded in an audit trail capability.

- Capturing Semantic Linkages: Develop an event-based model that allows one to *automatically* record events that are causally related to each other, and to chain sequences of events. Intuitively, events may be causally related if the same data is accessed from multiple locations (e.g., first on disk and later in memory).Whenever a casual relationship between entities accessing the same monitored objects is detected, for example, $E_i(O,L)$ by entity $p_1$ and $E_j(O^1,L^1)$ by $p_2$, the event model must record these linkages.

- Tamper Resistant Logging: Store captured semantic linkages in a structure we coined a "version-tree". Our initial idea for the version-tree is to create a modified B+−tree where the root node is indexed by the set of codepages belonging to the process accessing the digital object. The indices to the version-tree are the block numbers corresponding to the monitored object on disk, and the leaves are append-only entries of recorded operations on location *L*. The idea is to periodically write the version trees to disk in a tamper-resistant audit log where each record in the log is itself a version-tree.

- Origin Attestation: Develop efficient techniques for inferring what applications are accessing a monitored object. The developed tools will provide fuzzy fingerprints of applications, and allow for ACL-based decisions tied to application fingerprints. To support enhanced tracking and logging facilities, we will also create snapshots that capture all the process state, the virtual memory layout as well as all the code and data pages within the process. The data pages, for example, contain the buffers allocated on the heap by the process, while the code pages contain all the system modules needed to recreate that process at a later time — e.g., during forensic

reconstruction of what transpired during an access violation to a monitored object. The process fingerprints can be used to grant or deny access to the monitored object, and in that way, can be used as a form of discretionary access control.

- Lightweight Process Snapshots : This capability will be built using custom software that snapshots the memory contents. The snapshots will be stored in a popular cross-platform format that allows us to capture and annotate the entire process memory. These snapshots will also be incorporated into our provenance-aware audit trail, and provide further evidence of what actions the application was performing at the time of access to the monitored object. Specifically, the recorded snapshots will contain all the information required to recreate the offending process.

- Selective Blocking: Build a monitoring module that blocks and buffers packets that contain derivations of the monitored data. The idea is to identify offending packets by using the connection's 5-tuple (ip address and port) provided by the system call monitoring module. The entire connection is then buffered within the hypervisor and user notified of their action. We plan to send the notification via a driver built for Windows which translates the PID provided by the network module to a running application and presents the user with a popup message with the file name of the exfiltrated data, along with the offending application and time of access.

# 3 Methods, Assumptions and Procedures

Next, we review the work on specific aspects of the project.

## 3.1 Dynamic Provisioning

Dynamic provisioning allows for tracking resources at the file and block level. Our tracking system is initialized by a list of protected resources. A detailed description of these protected resources is necessary due to the multitude of ways Windows aliases devices and files. For example, a Windows 7 guest may use the following different descriptors to access the same file:

- \device\HardDisk1\SecureStorage\ProtectedFile
- \??\IDE\1\SecureStorage\ProtectedFile
- \??\C:\SecureStorage\ProtectedFile

### 3.1.1 Initial implementation

In the initial implementation we use a resource description that designates all files in a specified path to be "protected". The implementation thereafter is based on the system call tracking mechanism and will be further extended upon the completion of the disk tracking mechanism. When an application in the guest OS first accesses a protected file using either the *NtOpenFile* or *NtCreateFile* system call, it is added to a list of tainted processes, and all subsequent read and write requests are monitored. The *NtWriteFile* and *NtReadFile* calls are thereafter vetted for watch- listed applications. Special care is devoted to the **HANDLE** argument of those calls. The object referred to by the handle is extracted from the guest kernel using the **EPROCESS** structure, then compared with the protected object accessed with *NtOpenFile* or *NtCreateFile*. This procedure protects against attempts to circumvent the tracking mechanism using unconventional naming thanks to normalization of the object description to the tuple *(Device, Fully Qualified Path)*.

We maintain the information about watch-listed processes and protected files along with the list of blocks occupied by the resources so we can build the chain of custody as required in **(Task 4.4 Provenance Tracking)**. We are only tracking guest system calls, rather than using memory or disk block-level tracking, because dynamic provisioning is not yet fully implemented. Work on disk block-level data tracking is ongoing, which will provide the fine-grained information needed for the complete implementation. Figure 2 illustrates the Notepad text editor that opened a document from secure storage, then attempted to save it outside of the secure storage. The prototype implementation blocks the save. The default behavior of Windows prompts the user to save to their *My Documents* folder when permission is denied. Since this directory is also outside of secure storage, the user will not be allowed to save there either. Alternatively, we could inject a custom notification, as demonstrated in Task Selective Blocking.

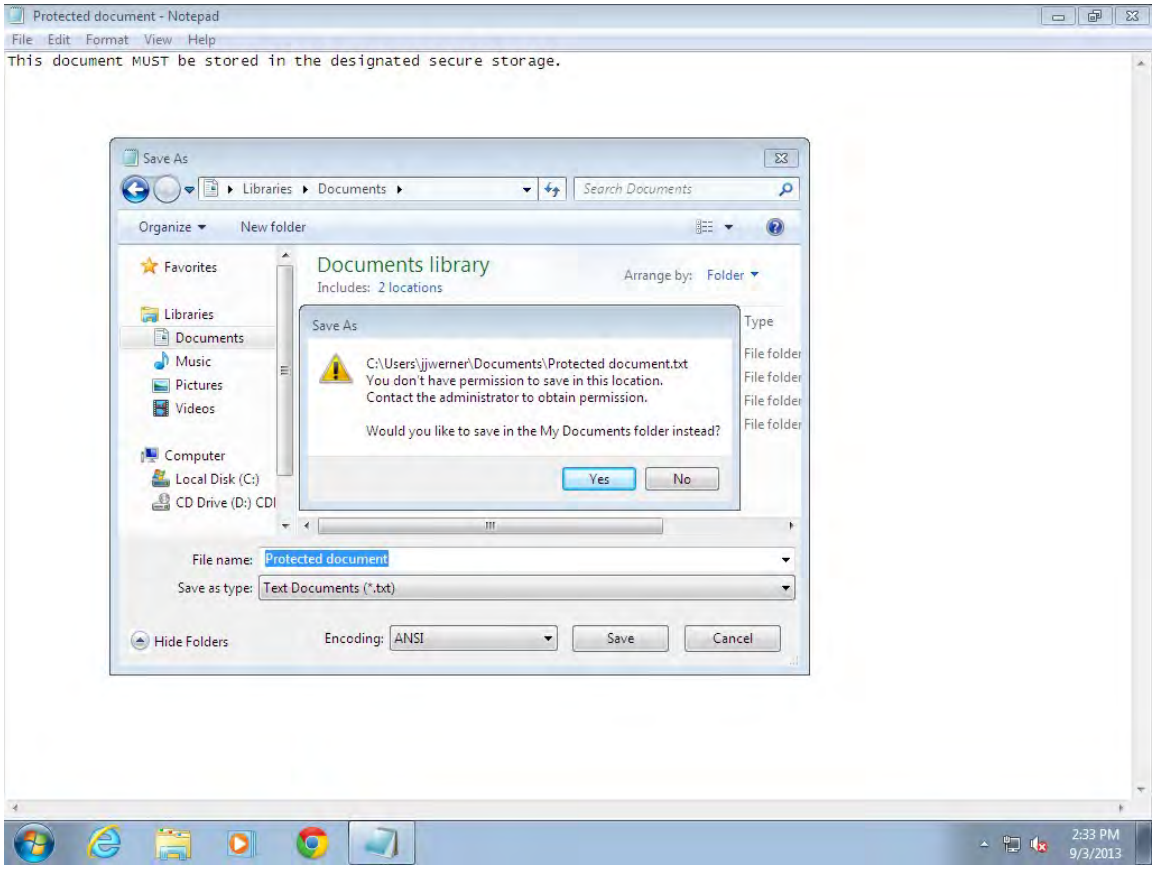


Figure 2: Blocked attempt to save a file

The initial implementation of the tracking mechanism could be circumvented using a low level disk access tool that makes block level copy of protected data. The complete implementation will prevent this attack vector by marking the resource as protected as soon as driver moves data blocks from disk into memory.

The goal of Dynamic Provisioning is to preserve semantic linkage across multiple data storages and to adequately describe protected resources. This allows for coexistence of both monitored and unmonitored data on same disk. We will use the following example in this section: a user is presented with a Virtual Machine with two logical disks, and one of the disks contains sensitive data in a file named *\datasets\patients.db*. Access to this file is monitored by hDLP.

For the purposes of analyzing the detailed breakdown of the input / output execution time we developed a tool to timestamp crucial events in the process execution.

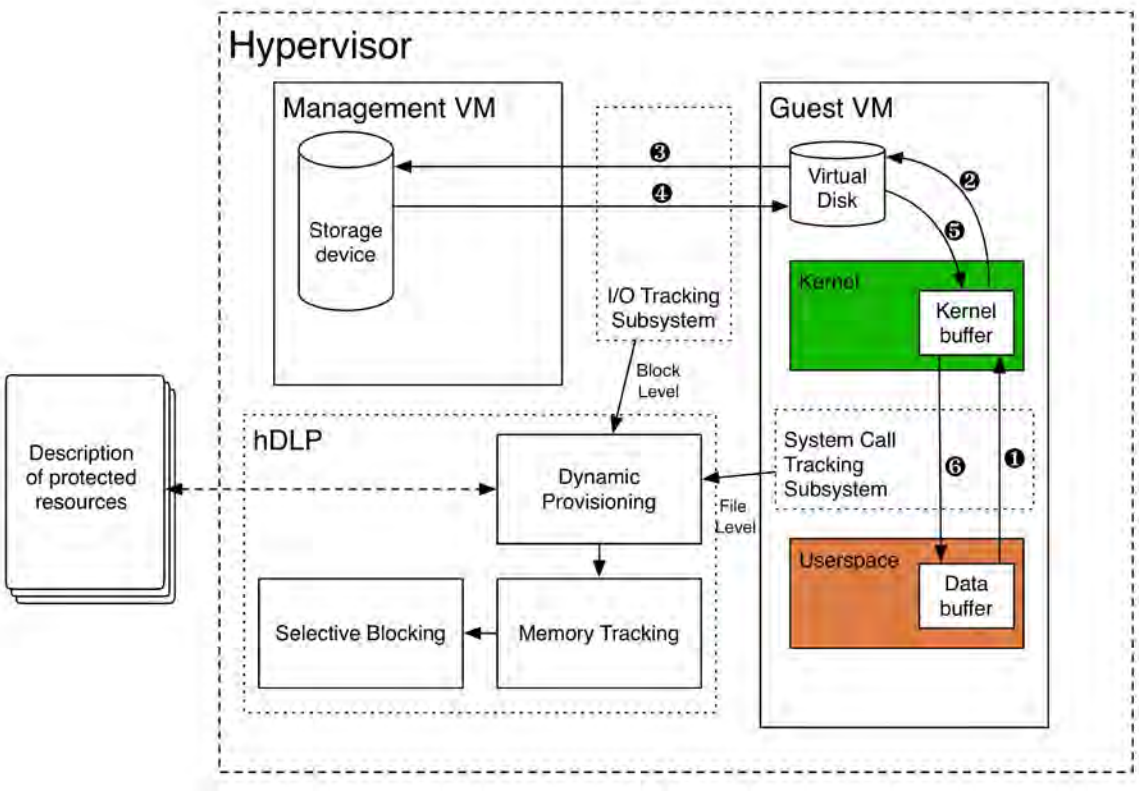


Figure 3: Data workflow within hDLP framework

### 3.1.2 Improvements to Dynamic Provisioning

In the following sections we describe two approaches: Block Level and File Level Dynamic Pro- visioning. Block Level Dynamic Provisioning tracks protected resources using the disk blocks as a unit of tracked data, whereas File Level Dynamic Provisioning directly tracks the contents of a file. The outcome of Dynamic Provisioning, a memory range that holds the sensitive data, is then provided to the memory tracking module and used for the selective blocking.

Dynamic provisioning requires a list of protected resources generated off-line and available to hDLP at its initialization time. An administrator initializing the DLP system is tasked with providing this list. The overall architecture of hDLP and data workflow is illustrated in Figure 3.

#### Block Level Dynamic Provisioning

Block level disk access tracking requires a list of blocks on the disk that store sensitive files. This removes the dependency on the guest OS kernel to provide any information, because raw disk access requests can be captured outside of the VM. Enabling block level disk access tracking requires the following components:

- A tool mapping the protected files to disk blocks to prepare a list of protected blocks.
- A hypervisor subsystem to track I/O requests from the guest domain.
- A modification to the memory tracking subsystem enabling tagging and untagging of the memory regions used for data transfers

### Mapping files to disk blocks

First, we implemented a tool that allows us to build mappings between protected files and the disk blocks they occupy. Such a mapping is necessary to bridge the semantic gap [1] in the hypervisor I/O system, which is not aware of the details of the underlying OS and its filesystem. The example file *\datasets\patients.db* that should protected by the hDLP will be represented as a tuple: <disk identifier, list of clusters>.

We built a parser for the New Technology File System (NTFS), the primary file system used in modern Microsoft Windows operating systems. This work uncovered a possible issue specific to NTFS that could pose a significant challenge for the block level disk access tracking. Logical files are usually stored in continuous allocation units called clusters. A file may occupy several clusters, with continuous clusters grouped into units called chains. Information about the files and their location on a disk is stored in special structure called Master File Table(MFT). If a file is small enough to fit within an MFT entry, the operating system will not allocate additional clusters for the file, but store the contents with the metadata instead.

To build a link between logical files and blocks on the disk, the tool first parses the partition and file system metadata. After processing that information, the tool can provide a bi-directional mapping between block and logical files. Additionally, we can extract the metadata and the contents of a file to perform further analysis.

### Tracking I/O requests from the hypervisor

An overview of a I/O request from the guest VM is depicted in Figure 3. We marked the steps that are important from the perspective of tracking VM requests and omitted the details of the Management VM provisioning the data, as they are not germane to the discussion at hand.

A request from a virtual machine to access a block of information from a storage device performs the following steps:

❶ A user reads the content of a protected file. A Data access request is issued using the read system call.
❷ The read request is formed into an I/O Request Packet (IRP) and sent to device driver responsible for instrumenting the data device. The device driver handles the IRP and sends a request specifying the disk identifier and list of clusters to the device.
❸ A virtual device driver in the hypervisor passes the request to a domain that provides the device access.
❹ The domain providing the data completes the request and notifies the guest domain via hypervisor notification channels that the request has been completed.
❺ The device driver receives the data and stores them in a kernel buffer
❻ The kernel of the guest transfers the data between the kernel structure and the userspace buffer and returns to userspace from the system call.

The envisioned block level tracking mechanism has to intercept the requests originating from the guest domain in step ❸, match the response in step ❹, and mark the buffer in the guest domain that holds the result of an I/O operation as tainted in step ❺. This entails tracking the copy operation from the kernel space to the userspace and the overhead associated with that operation in step ❻ Additionally, after step ❻ is completed the tracking mechanism has to remove the

taint tracking from the kernel structure.

Tagging and untagging tainted memory regions

Tracking the protected information using the block level requires tainting the data as soon as it is delivered to the guest virtual machine. This means that the kernel data buffers will be tainted and that the operation of copying the data from the kernel to userspace will require taint propagation. Kernel data buffers could have the taint removed as soon as the data is completely transferred to userspace, but this potentially leads to data leakage. A sound solution should remove the taint when the memory regions allocated to kernel buffer structures are overwritten with new content. The cost of this operation is significant: in the best case, tracking a copy of a 4kb data buffer would require 1024 exits to the hypervisor. Removing the taint information from the kernel buffer after the data has been moved to the userspace requires additional 1024 exits to the hypervisor.

### File Level Dynamic Provisioning

File Level Dynamic Provisioning is facilitated by data from the guest operating system: we leverage the information we obtain during system call invocations. The system call based I/O tracking is shown as steps ❶ and ❻ in Figure 3. A detailed, step by step procedure is shown in Figure 4. This method tracks file access related system calls: `open`, `close`, `read` and `write` to observe transfer of the contents of protected files between userspace programs. We require both parameters passed to the system call and all information returned by the kernel to minimize false positives in the memory tracking module. For illustrative purposes, let us assume that the example file *\datasets\patients.db* exists, it is 100kb long and a user issues a read request for 1mb of the file contents. If the memory tracking module operates only on the request information without the knowledge of the actual file size and the amount of bytes returned by the read operation, it will incorrectly mark 1mb region of memory as tainted, rather than only 100kb.

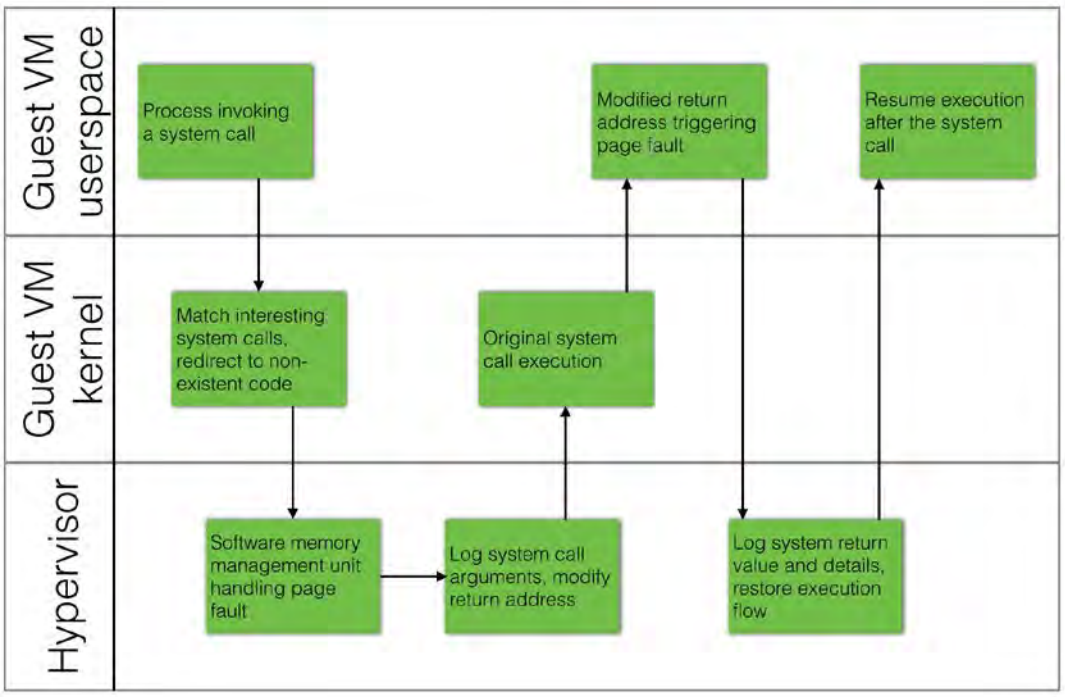


Figure 4: Control flow of system call tracking subsystem

To obtain the system call results, we modified the system call tracking modules to return to the hypervisor immediately after the system call completes. Recall the original architecture presented in Figure 13. To capture the system call arguments an exit to the hypervisor is triggered by causing a page fault on the known invalid memory location set as the Sysenter Machine Specific Register. Capturing the return value of the system call requires triggering another exit to the hypervisor. We achieve this task by modifying the saved system call return address[1] to force another page fault immediately after the system call return. During the second exit to the hypervisor we inspect the system call return value and the data structure holding the detailed results of the operation (such as the amount of bytes read for read system call, opened handle number for open system call). Afterwards, we return control to the instruction immediately after the system call return. The new architecture is shown in Figure 4.

Dynamic provisioning using the system call module allows us to track the userspace buffer after the guest kernel has already written to it. This means that tainting arbitrarily sized data buffers only requires two exits to the hypervisor.

[1] Stored on the userspace stack, stack pointer stored in the EDX register

**Takeaway**

The main benefit of the block level disk access is that tracking of data is possible without interfacing with the system call mechanism. In the current architecture, we trust the kernel of the protected system, assume an ability to detect an adversary tampering with the kernel integrity, and depend on the kernel data structures. Consequently, we argue that the benefit of not relying on the kernel information is not important in the current architecture.

We also argue that 2048 exits to the hypervisor to track a 4kb data buffer is prohibitively expensive compared to fixed amount of two exits for system call based approach. Additionally, the NTFS mechanism for storing small files creates a significant problem for tracking such files with block level granularity: accessing the metadata (for example: listing the contents of the folder containing tracked the file) will cause the memory tracking subsystem to erroneously track the flow of the metadata. This results in heavy overhead due to tracking unnecessary information.

## 3.2 Provenance Tracking and Capturing Semantic Linkages

The implementation of the memory access tracking extends the capabilities of Provenance Tracking and Capturing Semantic Linkages. Gathering fine-grained information about the accesses to process memory allows us to track data provenance and link the memory events to form a rich audit trail. We accomplish these tasks by intercepting the memory access from processes that are designated as tracked by the system call tracking component.

### 3.2.1 Capturing Semantic Linkages - Memory Tracking

The goal of the Capturing Semantic Linkages is to allow one to automatically record events that are causally related to each other, and to chain sequences of events. We initialize the chain of events by using the system call tracking mechanism to monitor filesystem related calls. When an application in the guest OS first accesses a protected file using either the *NtOpenFile or NtCreateFile* system call, it is added to a list of tracked processes, and all subsequent read and write requests are monitored. The destination argument of the *NtWriteFile* call, a buffer which receives data is used to initialize the memory tracking mechanism.

After obtaining the guest virtual address of the destination buffer, the memory tracking mechanism has to translate the in-guest memory location to an address that is recognizable by the memory management system of the hypervisor. The proposed system assumes that a software implementation of memory management shadow page tables is used and, that we can trap guest OS page faults. Shadow paging is a technique that creates a copy of guest page tables, sanitizes and propagates the changes between guest memory and hypervisor page tables, and allows the mem- ory management unit to use the hypervisor-vetted structures. The architecture of that solution is depicted in Figure 5. As we noted in section 4.1, the usage of shadow memory versus hardware assisted Extended Page Tables does not come at the cost of noticeable performance degradation.

The initialization of the memory location tracking is performed using the following steps:

**A1** Walk the hypervisor page tables to obtain the page table entries of the specific virtual address to be tracked.

**A2** Modify the page table entries to present the specified pages as not available.

**A3** Update the shadow page tables and flush the Translation Lookaside Buffer (TLB) to propagate the changes.

**A4** Add the guest page number and process CR3 to a list of tracked memory locations.

After the memory tracking mechanism has been initialized, any access to the memory region will cause a page fault and exit to hypervisor (VMEXIT). After this control transfer to the page fault trap, we execute the following steps:

**B1** Match the memory address that caused the fault to the list of tracked memory pages. If a match is found using the guest page number and process CR3 proceed to step two. Otherwise, pass the page fault exception back to the guest operating system.

**B2** Modify the page table entries to present the specified page as available, and propagate the changes.

**B3** Allow the guest OS to execute the instruction that caused the page fault.

**B4** If the offending instruction modifies other memory regions, add the newly modified page(s) to the list of tracked memory locations.

**B5** Repeat steps A2 and A3 marking the page table entries as unavailable, to catch subsequent accesses to protected region

This memory tracking mechanism improves the process of capturing semantic linkages as it provides a precise description of data manipulation happening between monitored system calls. For example, using the memory tracking mechanism, one can track a document and changes made to it from the initial load into memory until an attempt is made to save the document to disk or copy it over the network.

The currently implemented tracking mechanism is still in an early stage of development, and we are continuing to evaluate possible optimizations.

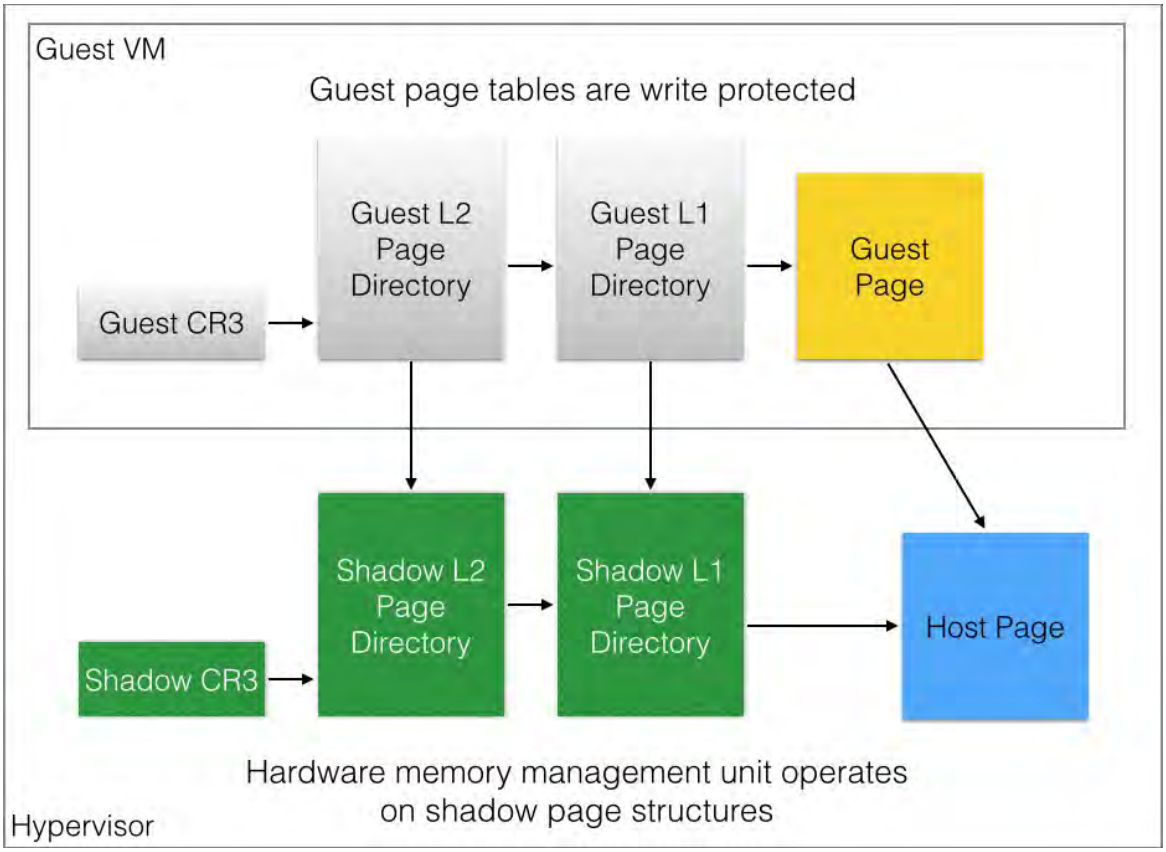


Figure 5: Architecture of Shadow Page Tables

### 3.2.2 Architecture dependent limitations

The implementation of the memory access tracking extends the capabilities of Provenance Tracking and Capturing Semantic Linkages. We are now tracking instruction-level accesses to process memory. We accomplish this task by intercepting memory access instructions from processes, interpreting the instruction arguments, and manipulating the page protections of the area accessed. In the implementation of instruction-level memory tracking we found a significant difference between the AMD and Intel virtualization extensions. Specifically, AMD CPUs lack the ability to single- step guest VM from the hypervisor, which we found crucial for efficiently handling instruction-level memory tracking. This development forces us to limit hDLP functionality on AMD-based platforms until a proper solution is found for this technical challenge.

### 3.2.3 Improvements to the memory tracking mechanism

We laid out the underlying mechanism for memory tracking in the previous section. In this section we focused on three specific patterns used in standard libraries to implement the copying of buffers in memory (i.e. the `memcpy` function). The specific implementation is chosen by the library based on the size of the buffer to be copied and its alignment on the page. We investigated the following implementations:

- Memory-to-register-to-memory using general purpose registers, which is used to copy small buffers.
- Memory-to-memory using string copy instruction, which is used to copy larger buffers not aligned on page boundaries.

- Memory-to-register-to-memory using SSE CPU extensions, which is used to copy large, page- aligned buffers.

Each of these operations consists of the set of assembly instructions. In general tracking the execution of a single instruction that caused a memory access violation (i.e. page fault) follows the steps (see Figure 6):

❶ Obtain source and destination addresses from the operands to the instruction that caused the page fault.[2]
❷ Remove the page protections.
❸ Allow execution of the instruction.
❹ After executing the instruction return to hypervisor.
❺ If the the source set is marked as tracked, mark the destination as tracked.
❻ Set proper page protections.

We evaluated the performance of our memory tracking mechanism using a worst-case usage scenario. The benchmark application loads a buffer marked as tracked and performs two memory- intensive operations: overwriting memory, and propagating tracked information to several subsequent buffers. Unfortunately, we observed five to thirty times slowdown of the benchmark application execution, which is in line with the state of art binary instrumentation [5].

In an effort to improve performance beyond the state of the art, we investigated the sources of overhead and proposed two possible optimizations. The above steps require significant processing overhead due to the fact that hypervisor is effectively single-stepping the guest VM through the memory access operations. Tracking a single instruction requires two exits to hypervisor: the first exit to allow access to memory (steps ❶ and ❷), and a second exit to restore memory protection (steps ❺, ❻,). Copying four bytes using memory-to-memory copy consists of a single instruction and thus causes two exits to hypervisor. Copy using registers requires two instructions: one to copy contents of the memory to a register, second to copy the contents of the register to another location in memory. Copying four bytes using a general purpose registers (32 bit registers), therefore requires four exits to hypervisor; copying sixteen bytes using SSE registers (128 bit registers) requires four exits to hypervisor.

[2] Depending on the type of the instruction, the operands may be either explicit (i.e. registers, memory addresses) or implicit (i.e. instruction takes no operands, but operates on well known registers). Instructions that copy contents between memory and registers use explicit operands, instruction that copy contents from memory to memory use implicit addresses: source in ESI register and destination in EDI register.

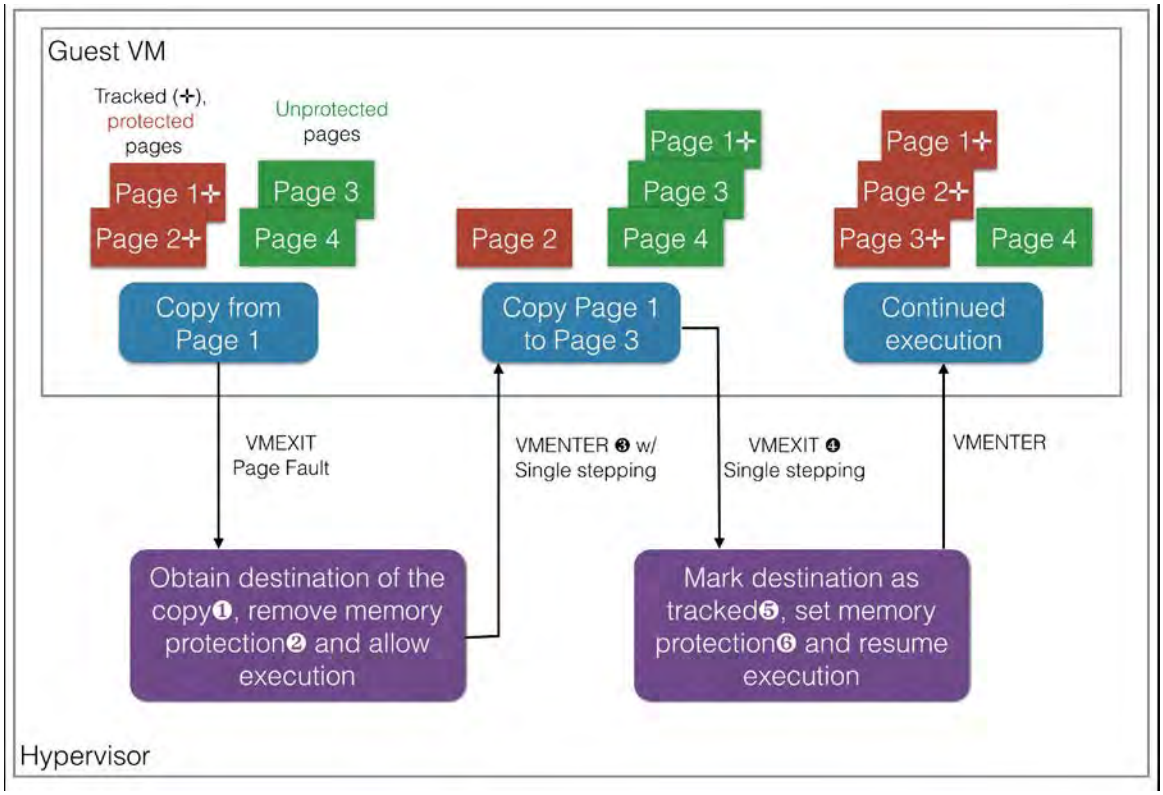


Figure 6: Instruction-level memory tracking

Based on those observations we are considering the following improvements that would allow for uninterrupted execution of certain memory access operations:

**O1** In the case of memory-to-memory copy, the number of copy instruction repetitions is con- trolled using the counter register (ECX). Instead of single-stepping through entire operation, one can set a breakpoint immediately following the copy operation, then reset memory protection after the operation is completed.

**O2** In the case of memory-to-register-to memory copy using SSE registers, one memory copy loop iteration consists of eight read memory interleaved with eight write memory instructions. One can allow uninterrupted execution of a single iteration of the loop by introducing a breakpoint after each loop round. Another possibility is to calculate the memory ranges accessed, set a breakpoint immediately following the copy operation and set memory protections after the loop has been completed.

Both optimizations presented above carry the risk that an adversary could prevent the system from properly tracking memory by altering control flow. We have not deployed these optimizations, as we are considering the trade-off between performance and potential loss of control over data, especially given that performance is already on par with state of the art.

### 3.2.4 hDLP Support for AMD Platform

In parallel with the work on memory tracking we introduced support for the AMD-based CPUs. There are some basic differences between the Intel and AMD processors that are easily reconciled, especially since both vendors provide virtualization support. However, the ability to single-step the guest VM from the hypervisor is present on Intel processors as a feature called Monitor Trap Flag (MTF), but has no counterpart on AMD CPUs. Lack of support for single-stepping in AMD CPUs limits the current implementation to system call tracking only for that platform. We considered workarounds for this issue; however, as it was low priority item we left this issue unsolved.

## 3.3 Tamper Resistant Logging

Once a workspace for a user has been provisioned, our goal is to provide a forensic provenance trail of the activities taken by that user involving the sensitive data. The basic idea is shown in Figure 7.

In the implementation of **Task** *Capturing Semantic Linkages* we built an infrastructure to capture causally related events. The goal of the *Tamper Resistant Logging* subsystem is to preserve and present that information to system operators. The core components of the hDLP platform built to support the auditing trail is shown in Figure 3. For completeness, the overall architecture of the system including the audit and reporting subsystems is presented in Figure 8.

Collecting and processing a potentially large number of events requires moving the information processing functionality from the hypervisor to the management domain and other machines. This approach allows for distributing the load to other machines saving the resources of the management domain. Generated reports provide detailed information about events, offer a time-line of related events and present state-of-the-art forensic information. The detailed architecture and capabilities of the reporting system are discussed below.

### 3.3.1 Architecture of the reporting system.

The Dynamic Provisioning and Memory tracking subsystems store the event information in the hypervisor memory. Since a running system may generate a substantial amount of information, event information should be promptly processed and evicted from the hypervisor.

We built a modular approach with dedicated subsystems for data acquisition and processing. Separating the two allows for offloading work from the Management Virtual Machine and for customization of the reporting system.

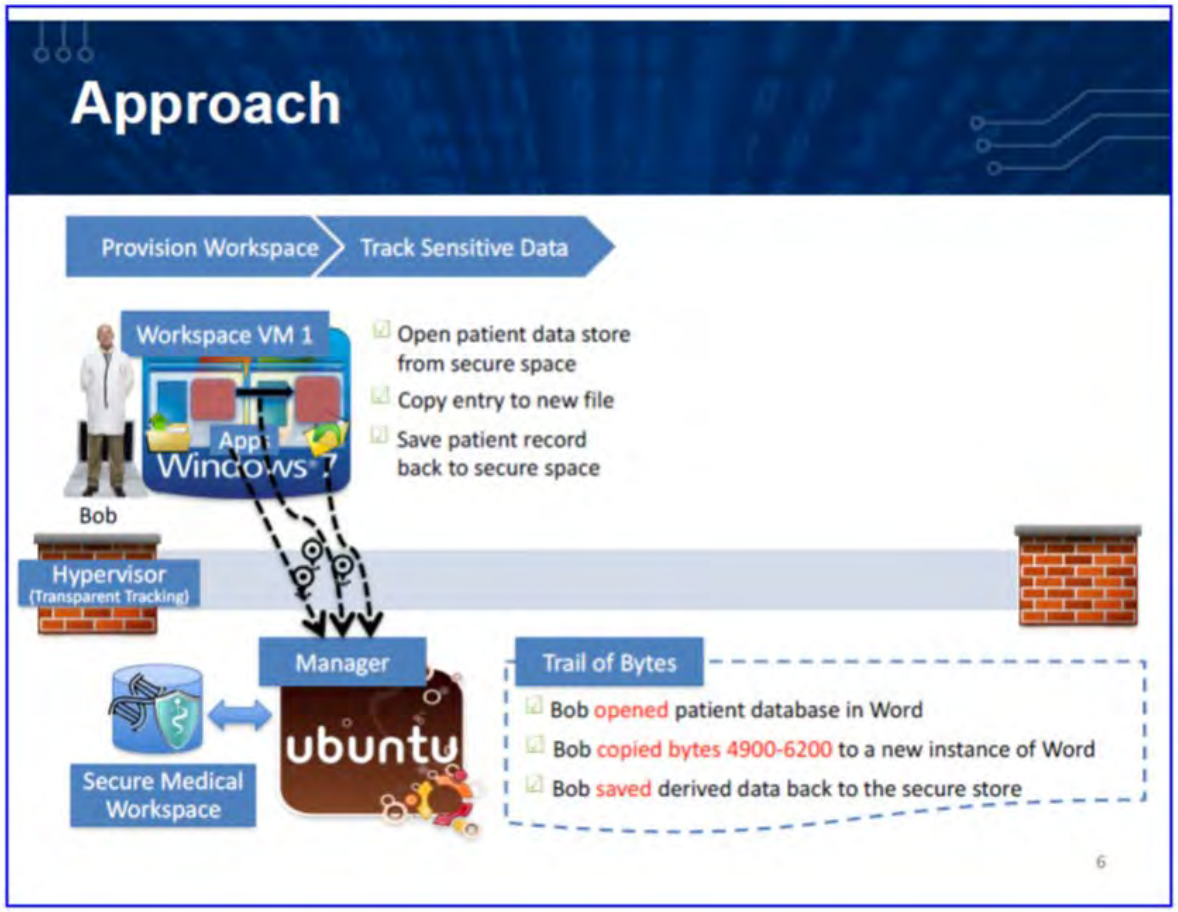


Figure 7: Provenance-aware audit trail

The data acquisition module of the audit mechanism is comprised of the following subsystems:

- A data collection service running in the management domain userspace.
- A data storage kernel module running in the management domain.
- A data export service running in the hypervisor.

We use a 3-tier architecture with the management domain kernel driver serving as an intermediate data location. This approach was dictated by a limitation of the Domain Control (DOMCTL) and memory copy mechanisms used to communicate between the management domain and the hypervisor. The hypervisor can access all the system memory using physical memory addressing, but it is not aware of the mapping of virtual to physical addresses in the management domain. This limits the amount of data that can be reliably copied in a data request to a single memory page (4kb) since userspace-allocated memory is not guaranteed to occupy contiguous physical pages.

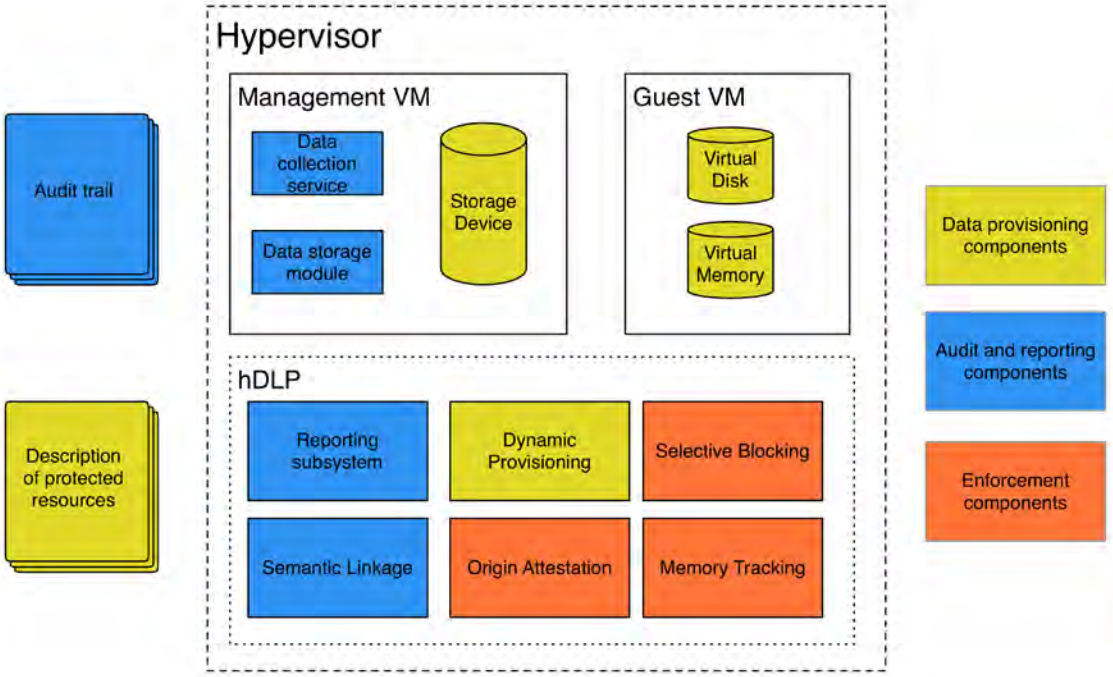


Figure 8: Architecture of the hDLP platform

The data storage kernel module solves this problem thanks to the guarantee of the kernel memory allocator allocating physically contiguous regions.

The userspace service polls the hypervisor for new events. When a new event is detected following steps are taken to extract the event information

❶ The data acquisition service queries the hypervisor for the event metadata.

❷ The hypervisor provides the event metadata, including the size of the event log entry.

❸ The userspace service requests a contiguous area of memory from the kernel to accommodate event log entry.

❹ The data acquisition service commands the hypervisor to copy the event details to an allocated kernel buffer.

❺ The userspace service retrieves the event details from the kernel using a dedicated character device.

After an event's information is retrieved from the hypervisor, its processing is handed to a reporting module residing in userspace.
Our reference implementation of the reporting module is built using the following components.

- An event data parser and report generator.
- A visualization tool for illustrating the data linkage.
- A web container hosting the information and providing a graphical user interface.

The above list could be modified to include a different user interface or presentation mechanism. We envision possible extensions for event data mining and even more

detailed forensic output.

A reporting module implements the following functionality:

- Update the event log with the event information.
- Correlate the data accessed in the event with reports of the previous activity
- Extract the forensic evidence (contents of accessed files, list of running processes and network connections) from the virtual data enclave.
- Generate an event report

The audit capabilities of the hDLP platform are presented in the results section.

## 3.4 Origin Attestation

Origin attestation, *i.e.*, determining which applications access a protected resource, enables us to restrict *how* information is accessed securely from the hypervisor. Discovering which applications are accessing a protect resource requires additional information from the guest. Our tracking mechanism requires information about both the running application and the current user. Gathering that information fills the semantic gap with respect to in-guest user credentials and processes. During its lifespan, every process can be uniquely distinguished using the value of a special-purpose CPU register called *Control Register 3* (**CR3**), which stores the location of each individual process memory page directory. A fingerprint of an application based on the snapshot of its process' code pages is a reliable way of identification; snapshots of the process memory can easily detect application modifications such as injected code. With the CR3 we are also able to extract detailed information about the application and the user by examining two in-guest data structures: the Process Environment Block (**PEB**) from guest userspace, and the **EPROCESS** structure from guest kernelspace. The process information from these structures, such as application name and login information, can be used to grant or deny access to protected resources.

As described in the previous section, process tracking is initiated when an attempt to access protected resource is detected. The hypervisor collects the following information about the process when it is added to tracked list:

- The owner information in the form of System Identifiers (**SID**).

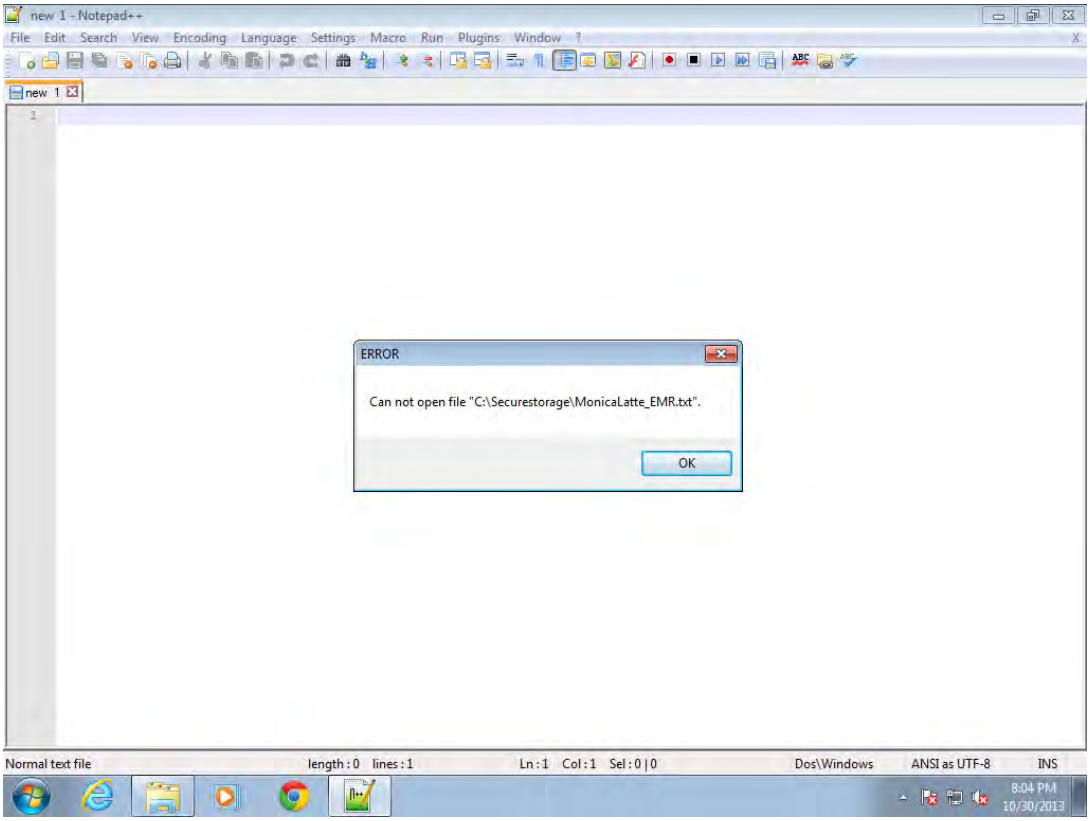


Figure 9: Blacklisted application denied access to protected resource

- The location of the PEB.
- The contents of the CR3.
- The name of the process executable image file.
- The base memory address at which process image is loaded.

Unfortunately, advancements in system defenses, namely Address Space Layout Randomization (**ASLR**), introduced additional difficulty in creating reliable application identifiers, but we are continuing work on fuzzy fingerprinting of application memory. Given the information we are collecting, we can currently distinguish between different applications reliably, but we cannot detect possible modifications made to them. In our prototype implementation we whitelist Notepad and Internet Explorer, meaning that only these applications can access files in the secure storage. Figure 9 demonstrates the denial of secure storage access for the Notepad++ application, while Figure 15 shows Internet Explorer successfully accessing secure storage. An additional issue that we addressed was removing a process from the watch list, after it ceased to exist. We currently have two ways of detecting process termination. First, we track the *NtExitProcess* system call and match the process signature with the tracked processes list. If a match is found, the entry is removed. Processes terminated using the *NtTerminateProcess* system call do not trigger that detection mechanism and have to be handled separately. To address this issue we attempt to detect when the shadow pages containing the process page table are dismantled. Unfortunately, this mechanism is still unreliable, and we are continuing our work on addressing this issue.

## 3.5 Lightweight Process Snapshots

For the most part, the currently available techniques for transferring files to and from VMs are all network based (e.g., Samba, FTP, SCP). Unfortunately, when using these methods the amount of time spent transferring memory snapshot data (typically hundreds of megabytes) far exceeds the amount of time needed run the analysis that generates the snapshot. Therefore, we required a more efficient means of moving data between Host and Guest virtual machines. **To address efficient data transfer, we designed a new system that transfers data from a Microsoft Windows guest directly to a Linux host.** In implementing this new subsystem we observed significantly bolstered performance of roughly 1.2 GB/s transfer speed. The key to achieving this level of performance is in directly sharing a physical memory region between the host and guest operating systems, rather than building on existing networking primitives.

In designing this new subsystem, we required:

1. Fast, bi-directional data transfer,
2. Transfers initiated by the host,
3. No in-guest networking or writes to disk,
4. A stock, unmodified hypervisor.

Bi-directional transfer enables us to quickly upload objects (to the Host) and extract memory snapshots. Host-only transfer initiation ensures a malicious program in the guest cannot cause a Denial of Service on the host by transferring large amounts of data. Eliminating the need for networking and disk write I/O in favor of shared physical memory is the primary mechanism for increased performance. Finally, using stock hypervisor software is a very limiting requirement, but as a matter of practicality it means our system is easy to deploy and maintain in existing environments.

Fortunately, Qemu/KVM has a built-in paravirtualized device implementation[3], called the ivshmem (inter-VM shared memory) device. The ivshmem device implementation attaches a virtual PCI card to the VM that performs no action other than to map a portion of the guest memory into a POSIX shared memory object on the (Linux) host. In other words, this enables the host and guest to directly read and write to the same small region of physical memory with zero overhead. However, ivshmem was originally developed for sharing data between Linux guests, while we require sharing between a Windows guest (i.e., where the SMW applications reside) and the host. Nonetheless, the availability of this shared memory primitive enabled us to build our data transfer mechanism from the bottom-up.

### 3.5.1 Technical Details

We developed a Windows kernel-mode driver that interfaces Windows user-mode programs with the shared memory. The driver provides a file-like interface, wherein a guest user program uses the CreateFile API to open the special device handle, then WriteFile or ReadFile APIs to read or write data to the shared memory region. When writing a file from guest to host, for example, the driver divides the data provided

[3] A software device that a VM-aware guest can use to efficiently communicate with an underlying physical device

by WriteFile into chunks that fit into the shared memory region and sequentially writes each chunk after the guest has processed the previous chunk. We synchronize the transfer between host and guest by polling and updating status flags in the shared memory.

To initiate data transfers from the host-side we take advantage of another built-in Qemu/KVM feature: the ability to artificially inject key-presses to the guest. Using this feature, a host-side user program programmatically injects a special key-press combination that triggers an event in a service we run within the Windows guest. Once the guest has been notified, it follows our custom protocol for exchanging information (through the shared memory) indicating the action the guest service should perform. The actions include read, write and execute, with auxiliary information exchanged that indicates filenames to read or write, or the command to execute. We note that command execution was not an original design goal. However, its implementation using our data transfer design was trivial and it benefits our analysis by faster execution of commands and completely removing reliance on host-guest networking.

### 3.5.2 Performance Analysis

Host-guest data transfers can have a significant impact on our overall design. Our improved snapshotting and transfer facilities show than we can alleviate most of the performance concerns that plagued our earlier prototype. Our recent experimental results (see Figure 10) show that we are able to achieve roughly 1255 MB/s — **i.e., more than a 100 times increase in speed over our prior approach**.

## 3.6 Selective Blocking

Recall that our notification mechanism for alerting users of the secure workplace of potential access violations serves two roles: first, it serves as a mechanism for educating the user about the offending action to prevent similar offenses in future, and second, it provides a way for user to request a policy override to continue the (possibly non offending) action.

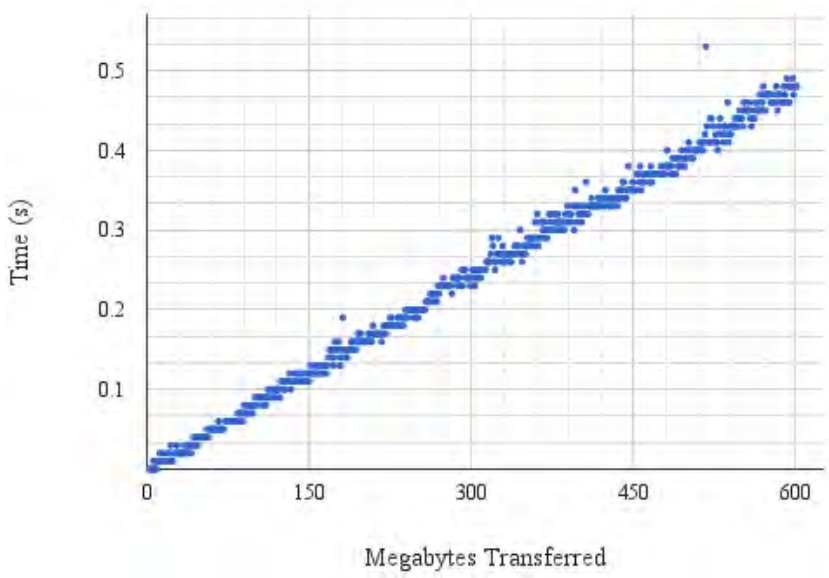


Figure 10: Speed of transferring varying sized objects between the Guest and Host Machines

In the design of the notification system, we required:

1. an approach that does not reside in the guest VM,
2. a mechanism that allows for bi-directional notification,
3. the ability to suspend the process with the offending action.

Removing the notification mechanism from the guest VM eliminates the necessity of modification of the deployed appliance and protects from malicious tampering with the mechanism. Two way communication allows for end user input and possible override mechanism when the action marked as offending is certified as benign by the user. Suspending the offending process is necessary to avoid data ex-filtration while notification and user reply are delivered.

To better address the requirements of the notification mechanism, we first investigated methods of process control flow redirection within the guest machine and evaluated their portability and applicability in the hypervisor. We currently consider two separate approaches and their further evaluation is dependent on the progress with the task of porting preliminary prototype to KVM (**Task 4.2**) and updating the porting the preliminary prototype to current version of Xen hypervisor.

The experiments for the evaluation of the control flow redirection allowed us to develop in guest prototype for object tracking. The in guest solution is applicable only for a model where the users are not malicious but may unintentionally try to access protected information. The ease of integration (no changes required to management infrastructure, trivial registry change and addition of library in the deployed appliance) allows for rapid deployment and collection of preliminary results in the Secure Medical Workshop (SMW). We also envision that usage patterns collected in this stage should allow us to improve our understanding of how to best tackle design decisions for our dynamic provisioning (**Task 4.3**), provenance tracking (**Task 4.4**), and capturing semantic linkage (**Task 4.5**) implementation.

### 3.6.1 Technical Details

In the process of exploring the notification mechanisms we tested and implemented three different methods of the application control flow redirection. Using the Windows API we can modify the application behavior in three different ways:

1. Injecting code into a running thread
2. Creating a thread inside a running process
3. Injecting a Dynamically Linked Library (DLL)

We found that strengths and limitations of each approach significantly change after changing the scope of the experiment and using them in the context of hypervisor. Our analysis show that the injection of the DLL is the best venue for the in-guest solution for user notification and data usage collection. The method of injecting code into a running thread has the most severe limitation while running in the guest VM, however its applicability is dramatically improved when applied from the hypervisor. The method of injecting a thread into a running process has limited application while used in the guest VM, and does not provide any additional benefits in the context of hypervisor. Thus we focus on the first and third options as potential solutions.

#### DLL injection

In what follows, we briefly highlight the approach we took for DLL injection. *We note that while DLL injection method does not satisfy the requirements of the notification system for the final deliverable, it serves as an important step in understanding the guest machine intricacies and allows for a preliminary implementation for testing and usage patterns collection.* Proposed in guest solution is tasked with tracking the application calls to existing native APIs, collecting the information about their object access (eg files, directories, network sockets) and validating the requests against a security policy. The implementation leverages a technique of Windows API called Hooks. Specifically, we use the SetWindowsHookEx() which creates a system wide hook that eventually is executed in the address space of each hooked process. Within the DLLmain function (which is done once during the injection into a particular process), we read in a policy file.

At present, we use a simple policy format that specifies what applications (by name) are allowed to perform which actions (read, write) in what directories. If the current process does not have any policy statements applicable to it, its functions will not be wrapped. An intercepted CreateFile() Windows API call then checks against all relevant policy statements and will either return an error (”access denied”) or will proceed to call the native API CreateFile() function and return its results. In addition to the policy checking, we verify that the executing process is binary that is allowed to run on the machine by calculating a cryptographic hash of the binary and comparing that against known and allowed signatures. A notification sent to user resulting from the policy enforcement is shown in Figure 11.


Figure 11: A sample notification with user input option using Windows API

At this stage, we are experiencing some difficulty with interacting between 32 and 64bit processes, and plan on addressing this issue. At present, our current implementation is designed for monitoring and controlling 32 bit processes in 32 or 64 bit environment (native or Windows 32-bit on Windows 64bit (WoW64)).

### Injecting code into a running thread

The in-guest code injection into a thread is achieved using Windows API OpenThread(), WritePro- cessMemory() and SetThreadContext() calls. The targeted process is redirected to an injected code by implicitly changing its control flow via the modification of the instruction pointer. That said, such an approach is volatile and provides no way of receiving feedback from the injected process; it may, however, be used to stage a DLL injection startup. We anticipate that when used in the context of hypervisor, this approach will provide bidirectional communication with the controlled process. A sequence diagram of the code injection into a running thread from hypervisor is depicted in Figure 12.

In hopes of leveraging existing work, we first experimented with an implementation called Ether that enables Windows XP system call interception at the Xen hypervisor layer. At its creation time (2008), it was written for the Xen API version 3.1.0. However, since its inception, the API has evolved and changed drastically, and made porting of the existing Xen code challenging.

We faced similar issue with our prototype that was initially implemented for Xen API version 3.4. Therefore, we opted to instead take a clean room approach to directly implement on Xen API version 4.3 or port to KVM. The advancements in the CPUs supporting virtualization also forced us to re-evaluate the methods of detecting events in guest virtual machines. Both AMD and Intel CPUs offer hardware assisted memory paging, that when used renders previous approach to event detection challenging. While it may be possible to revert to using software-only paging mechanism called shadow paging, and our initial assessment of performance penalty shows no or minimum impact.

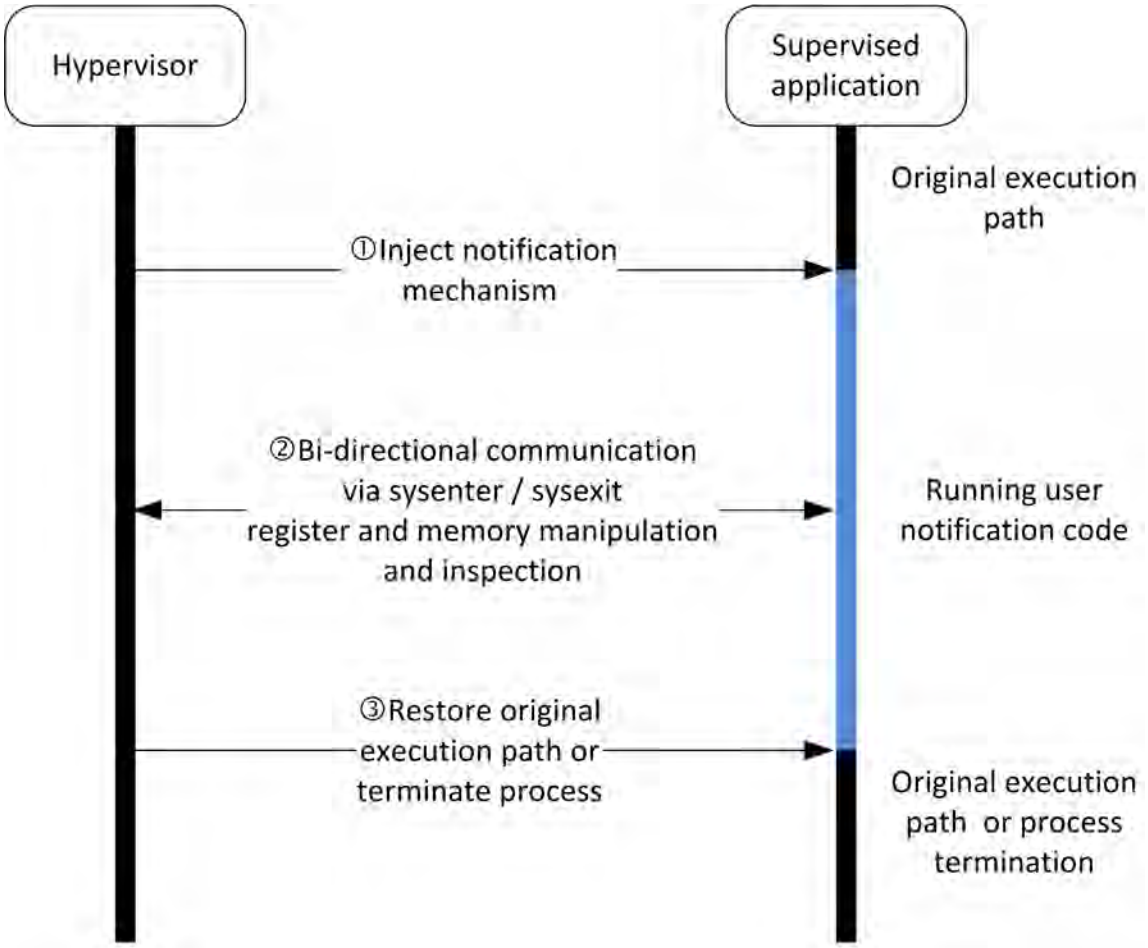


Figure 12: A control flow diagram of hypervisor injecting code into in-guest process

### 3.6.2 Hypervisor based Selective Blocking

We discussed the requirements and limitations of the notification system in the previous sections and also implemented an in-guest proof of concept that satisfied all but one requirement: it was permanently resident in-guest instead of in the hypervisor. After investigating userspace solution we focused on both moving this notification into the hypervisor and implementing monitoring facilities within the hypervisor that trigger the notification. Our goal was to minimize the in-guest footprint at all stages, which eliminates the need to modify the guest operating system and protects from malicious tampering of the monitoring and notification mechanisms.

In summary, we (1) implemented system call monitoring in the Xen hypervisor, (2) adapted in-guest thread injection-based notification and selective blocking mechanism to operate inside the Xen hypervisor, and (3) further evaluated hypervisor performance in two modes of operation, the results of which affect future design decisions.

The experience with the in-guest solution gave us valuable insight to many of the challenges of the notification mechanism. We continued the transition to a hypervisor-based approach to move notification and response logic completely out of the guest Virtual Machine. With (1) and implemented, the notification and response is ephemeral - code is injected into the guest when triggered by a system call, the user responses via an injected dialog box, then the injected code is

removed without leaving any traces in the guest. In (3), we evaluated guest performance with and without hardware assisted *Extended Page Tables* (EPT). Enabling the EPT feature is intended to increase performance by allowing page-fault events to be handled in-guest rather than by the hypervisor. Several guest introspection techniques, such as memory tracking, rely on the hypervisor receiving page-fault events. Thus, if EPT significantly improves performance it warrants further research into alternative techniques for guest memory tracking. However, our earlier benchmarks did not show a significant benefit of the hardware assisted solution, and we found confirmation of our results in [6]. Therefore, at this point, we see no benefit in enabling or supporting the EPT feature.

### 3.6.3 Implementation Details

Moving away from an in-guest solution towards the hypervisor introduces a problem called the *semantic gap*—that is, within the guest we can make use of OS-provided facilities for hooking and understanding events at a fine-grained level, as well as interacting with the user. However, from the hypervisor perspective, guest activity is more coarsely composed of a sequence of faults and device I/O requests. The guest semantics of events at the hypervisor-level are not immediately apparent. In implementing (1) and (2) in the hypervisor, we have to bridge the semantic gap by first detecting guest events that require selective blocking, then introducing notification code into the guest to interact with the user. We accomplish these tasks through hypervisor-based system call tracking and code injection similar to in-guest solution that injects code into a running thread.

#### Event (system call) detection

By default, a hypervisor will handle a configurable subset of various fault-types generated by the guest. For example, with EPT disabled the hypervisor will handle a fault generated when the guest accesses an invalid section of memory. Other faults include division-by-zero, debugging instructions, and software and hardware interrupts, etc. However, there is no specific system call fault, which highlights part of the semantic gap problem. Older operating systems used a software interrupt convention for system calls that would generate a fault observable by the hypervisor. Unfortunately, modern systems use a dedicated instruction: sysenter (or syscall on 64-bit architectures), which does not generate a fault observable by the hypervisor. Instead, the sysenter instruction directs the CPU to lookup the system call handler at the address specified by a special *Machine Specific Register* (MSR). The MSR is initialized with the address of a valid system call table by the guest OS when it boots.

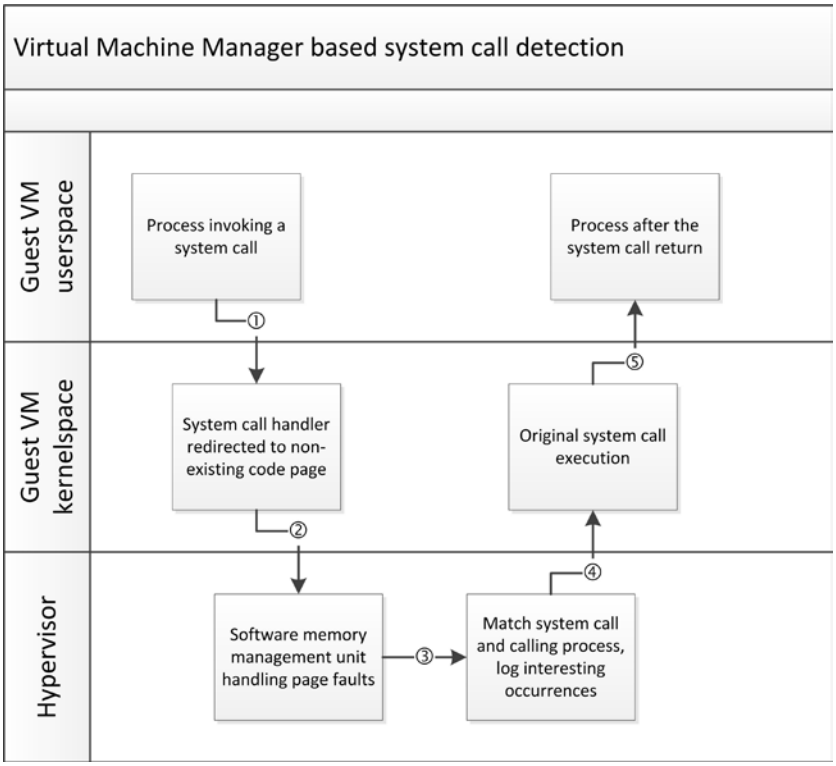


Figure 13: System call detection mechanism

To detect system calls in the hypervisor we must force a fault to be generated that is observable by the hypervisor. One straightforward way of doing this is by altering the value stored in the MSR. The technique we use is to set the MSR to a location in memory that does not exist (*e.g.*, as proposed in [2]). As a result, a page fault will be generated for every system call, which the hypervisor can monitor. Figure 13 illustrates the logical flow of control when a system call is made with the MSR mechanism in-place. This technique is portable across 32-bit and 64-bit Microsoft Windows and Linux operating systems with trivial modifications.

In addition to system call detection, we also need to determine which system call was made (*e.g.*, file operations, networking, graphics, etc.), as well as the parameters of the call (*e.g.*, the file name, type of access, etc.). In general, system call and parameter information is accessed by following pointers in CPU registers to access guest memory locations that contain the information. However, the details vary across both OS and OS version, as well as the system architecture. Our implementation in the Xen hypervisor supports 32-bit Microsoft Windows 7. To demonstrate system call detection, trigger notification, and selective blocking, we implemented support for filesystem-related system calls. Specifically, we trigger notification from the hypervisor when a guest user attempts to copy a file from the guest OS to an external USB device. We will extend this later to support more complex policies from the hypervisor.

#### Notification and selective blocking mechanism

As alluded to in the last section, detection of a system call sometimes triggers notification of an offending action within the guest. For example, copying a protected file to an external USB device is detected by observing a series of system calls in the hypervisor that read in a file, then write it to the USB device. When this occurs, our goal is to notify the user that sensitive information is being exfiltrated and enable them to decide whether to allow the action. In this context, *selective blocking* has two meanings. We select which system calls trigger a notification (or blocking action), but we also empower the user with selecting whether their action should continue. Policy defines which actions a user may override, but in the current implementation we always allow user selection.

We implement selective blocking, including interactive user notification, by immediately redirecting the execution of an in-guest process that violates the USB filesystem policy. To achieve that, we introduce code in the guest VM when triggered by a system call in violation of policy, force the guest CPU to execute the injected code while pausing the original thread, wait for the user to allow or deny the action, and finally retrieve the results generating from user interaction with the injected dialog. Figure 14 illustrates the overall workflow of selective blocking from the hypervisor.

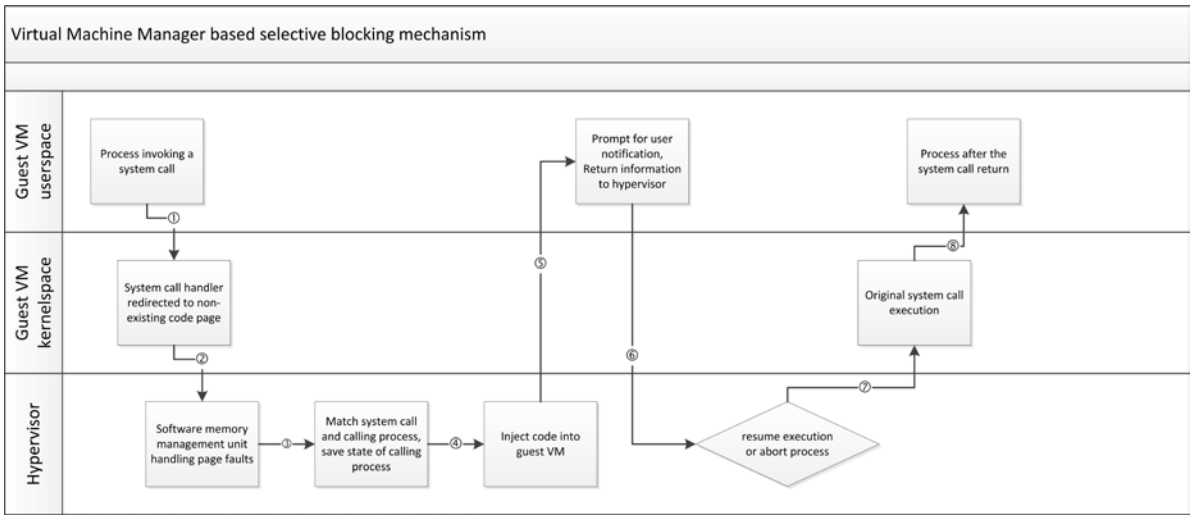


Figure 14: Hypervisor based injection of notification mechanism

There is, however, one caveat with our approach of injecting code into the guest. The thread we inject code into is totally unaware of what has happened, as it is paused during execution of the injected notification dialog. Further, it fully expects that the execution environment (*e.g.*, registers and memory) remains consistent before and after a system call. Therefore, we must ensure that the selective blocking mechanism preserves the integrity of the guest execution environment. That is, anything we put into the guest must come out when notification is done, and the original guest program state must be restored. To achieve that requirement we first save the state of the process - *e.g.*, we create a complete copy of the CPU registers and memory that will be modified by the injected code. After returning from notification, all the changes are reverted and the in- guest application is resumed at the instruction that triggered the policy check. If the offending application should be terminated (based on policy or the user input), the hypervisor introduces another block of code to gracefully terminate the offending process.

# 4 Results and Discussion

## 4.1 First Technology Preview

The system call tracking and selective blocking mechanisms implemented in the early stage of the project have limited utility without detailed information provided by the origin attestation and dynamic provisioning. After we built those new components, we were able to formulate and enforce some simple security policies. For example,

- We can deny users with specified System Identifiers (**SID**) access to data in specified protected locations.
- We can deny a specified application access to data in specified protected locations.
- We can allow access to specified protected location or resources, but deny the ability to extract files from protected locations.

The implementation of the enforcement mechanism has been throughly tested with the Microsoft Windows 7 operating system, which is currently the most prevalent guest OS platform.

## 4.2 Selective Blocking Test Results

While testing the selective blocking mechanism we found and corrected conditions where the selective blocking mechanism could be invoked multiple times or overwrite the original saved code if invoked from different threads of single process. We also found that the selective blocking mechanism may cause issues in multi-cpu guests when the OS running on other cores finds inconsistencies caused by the selective blocking mechanism in other threads. We will investigate this issue further, for now, our prototype avoids such problems by running guests on single core only.

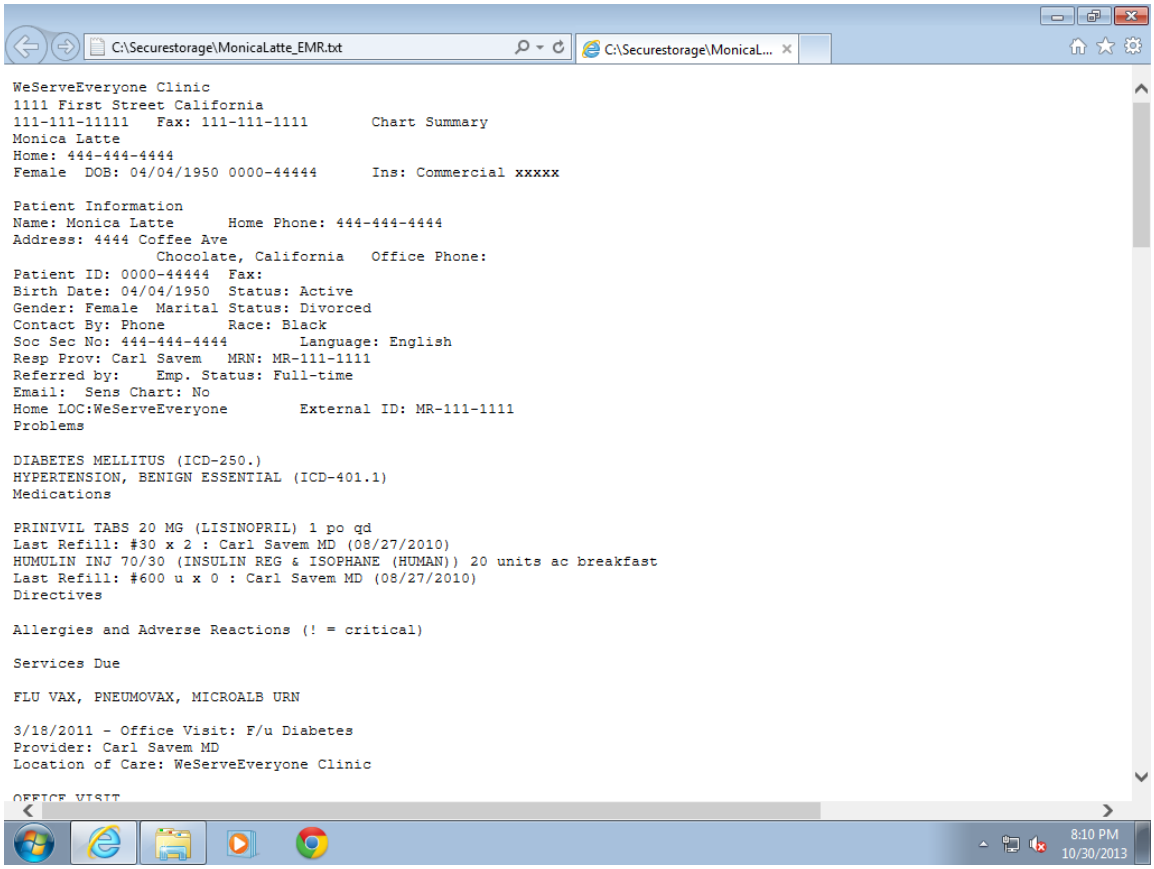


Figure 15: Whitelisted application granted access to protected resource

## 4.3 Pilot deployment of the Hypervisor-based DLP in SMW

**Secure Medical Workspace:** We remind the reader that the Secure Medical Workspace (SMW) [4] is intended to be a comprehensive solution to the data security challenges that arise in provisioning clinical data for research purposes. The SMW was developed jointly by RENCI and UNCs NC Tracs Institute and motivated by requirements at both UNC and Duke University. In translational and clinical research, patient data is often provisioned to researchers through sanctioned mechanisms, such as on encrypted file systems residing behind institutional firewalls or through secured email. However, once provisioned a significant security risk exists due to the inadvertent or purposeful exposure of data due to lost or stolen portable storage devices (e.g., laptop computers, USB drives) or hard copies of data that had been printed. This security risk can render ineffective methods in place to safeguard data in institutional repositories. **Our prototype supports data provenance in the SMW environment.**

The SMW model for provisioning of research data was developed as a solution to this data leakage problem. In this model, provisioned research data is provided to researchers on virtual machines that allow for monitoring and controlling the movement of data out of the virtual environment by sophisticated data leakage technologies. The SMW is currently operational at UNC and being used for collaborations with external entities and for internal studies.

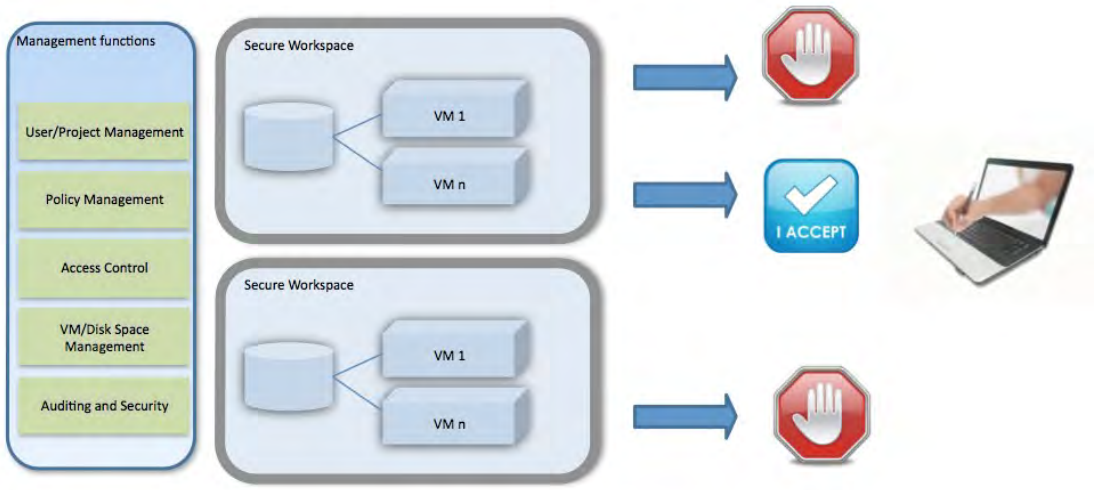


Figure 16: Secure Medical Workspace Overview Significant security features of the SMW include:

1. Virtualization technologies to facilitate the set-up, data provisioning, management, and tear- down of protected virtual workspaces;

2. The ability to provide clean virtual desktop images in order to mitigate the possibility that viruses and malware may expose data to the outside world;

3. Endpoint Data Loss Prevention (DLP) technologies and techniques to prevent unauthorized use and/or transmission of data, in order to maintain compliance with University policies regarding Sensitive Information;

4. The ability to require investigators to acknowledge that any removed data, such as figures and graphs, do not contain sensitive data;

5. Standard set of operating systems and tool sets (golden images) that allow support staff to focus on hardening those technologies and to rapidly deploy them;

6. Remote access technologies to further isolate the data; all work (e.g., analysis) involving sensitive data is confined to the SMW; the researcher is able to manipulate the data but can remove, copy, or print sensitive data from the SMW only based on institutional policies;

7. The ability of compliance officers to review and audit data usage and movements to ensure institutional policies are followed.

A key requirement is that the SMW environment must present a challenge to the user at the point in time that they attempt to remove data from the workspace and the SMW environment must allow the user to override the environment (with justification and auditing) when removal of data is time sensitive and critical to business requirements. These requirements are currently met by the Endpoint version of DLP, and *our prototype significantly extends* the capabilities it offers.

In collaboration with our partners at RENCI, we deployed the first version of the hypervisor DLP system to verify its usability, stability and performance. The deployment is capable of tracking and limiting file access, while enforcing the same security policy as the one used by the Secure Medical Workflow commercial

DLP product. The policy prevents modifying the data on the non- local volumes such as network shares and remote desktop drives. The team, lead by co-PI Schmidt and consisting of three faculty members and one grad student, is performing the evaluation of the system. Hypervisor based DLP is currently deployed on two machines equipped with a single quad core Intel i7 CPU and 16 GB of RAM, which each host up to four virtual machines.

### 4.3.1 Initial User Feedback

The first subjective benchmark relies on user feedback about the system responsiveness and the general experience of using VMs on the hypervisor-DLP enabled host. One of the users reported seemingly slower download speeds, and another mentioned laggy responsiveness when using a Re- mote Desktop Protocol connection. Other users claimed no visible slowdown of the DLP-enabled virtual machines. Some initial reports of performance issues were found to have stemmed from VMs that were improperly configured and not running paravirtualized drivers. The reported problem of slow download speeds was most probably caused not by the DLP, but by the network activity of other guest VMs. In our opinion, the subjective benchmark and some of the observations may be skewed, since the users compared the speed and responsiveness of the local bare metal machine versus remote desktop connection. Performing a blind test with a virtual machines with and without the DLP solution could dispel some of the perceptions of the slower performance. The subjective analysis is still underway, and these shortcomings in testing will hopefully be resolved.

## 4.4 Objective Benchmarks of the System-call Tracking

The quantitative evaluation of the system performance was conducted using the SPEC CPU2006 benchmark suite[4]. The benchmark suites measure the system performance when carrying out certain computationally intensive tasks. We expect that protected systems will be needed for statistical analysis, image analysis and compiling software. We chose the suite of integer benchmarks CINT2006, because it mimics some of these expected usages. We found that the system call tracking mechanism in the hypervisor introduces a 5-10 percent performance penalty. The penalty is closely associated with the number of context switches to the hypervisor (VMEXIT events) which occur on every system call to the guest operating system.

[4] Available at: https://www.spec.org/cpu2006/

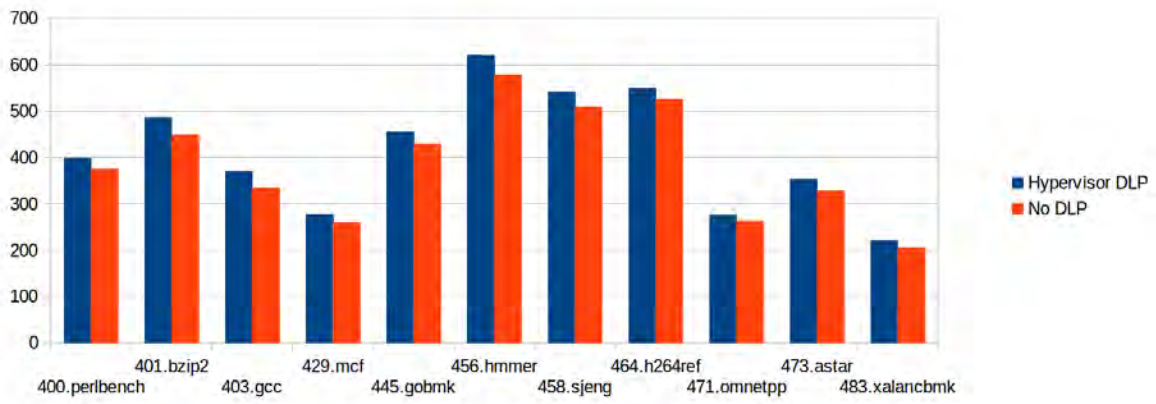


Figure 17: Comparison of SPEC CINT2006 results

We compared the median run-times of the benchmarks on the machine running our hypervisor- DLP solution with the Xen hypervisor with the hypervisor-DLP codepath disabled. The gathered results are presented in Figure 17.

We expect that the performance results may differ for other task loads. A benchmark of system call execution time would show much more overhead, but we expect that applications used in our targeted deployment will spend most of the time in the users-pace with only occasional system calls.

## 4.5 System-call Tracking Performance Improvements

The quantitative evaluation of the system performance was conducted using the SPEC CPU2006 benchmark suite [3]. The benchmark suites measure the system performance when carrying out certain computationally intensive tasks. In early quarters, we chose the suite of integer benchmarks CINT2006, and we used the same benchmarks to measure the improvements. Based on the observations from previous quarters we focused on reducing the amount of exits to hypervisor. We introduced an optimization that moves system call tracking pre-selection to the guest virtual ma- chine. Previously, all system calls caused an exit to hypervisor but only a select few (`open`, `read`, `write`, `exit`) were further processed. Other system calls, which are not a concern for DLP, still caused significant overhead. To remove these unnecessary VMEXITs, we introduced a system call pre-selection module that runs in the context of the virtual machine. The pre-selection module checks the system call to see if it is one of the list, if so the module triggers an exit to hypervisor; otherwise, it returns to the guest OS system call handler. The pre-selection module runs using virtual machine enlightenment, a communication mechanism between the guest virtual machines and a hypervisor, which provides integrity protection preventing a potential adversary from tampering with the code.

We compared the median runtimes of the benchmarks under the original, un-optimized hypervisor-

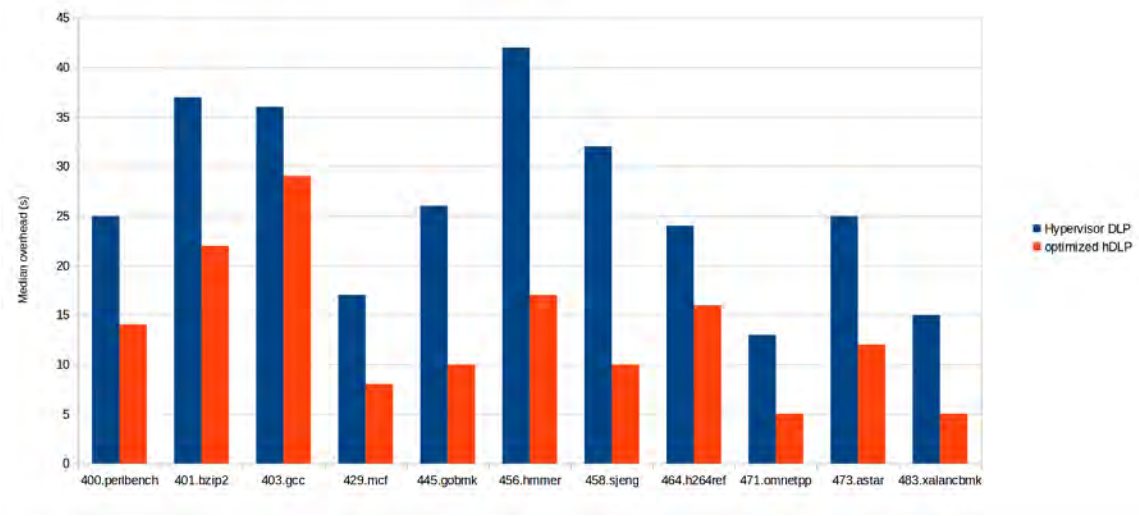


Figure 18: Improvements of SPEC CINT2006 results with hypervisor optimizations

DLP against the newly optimized hypervisor-DLP. The optimizations reduced the overhead of the DLP mechanism across all the benchmarks, in several cases by over 50 percent (see Figure 18). The improved system call tracking mechanism introduces 2-8 percent performance penalty versus 5-10 percent of the previous release.

## 4.6 Tools for profiling Windows Kernel I/O Request Packets

Our partners at the University of Michigan provided us with tools to capture the traces of accesses to objects, such as sensitive files, from the in-guest perspective. The objective of this tool is to present a detailed breakdown of data access time, to find possible bottlenecks, and improve the performance of hDLP. We were interested in capturing the following data points:

- Time spent in the kernel per system call
- Time spent completing an I/O operation

Measurement of the time spent in the kernel is performed by hooking the entry and exit points of the system calls in the *ntdll.dll* library. The time spent completing an I/O operation is measured by a custom kernel module called minifilter that intercepts all I/O request packets (IRPs) in the kernel before they are sent to the device driver. Both observations are synchronized thanks to the timer that is shared between userspace and the kernel. An example output of the tool recording start and end timestamps of **CREATE** and **WRITE** operations from the kernel module named `ntfs_tracer` and from the userspace program named `apihijacking` is presented below:

```
40.06158829 ntfs_tracer: pid=6d0 op=CREATE start=168827052445 end=168827052860

40.06168747 [1744] apihijacking: pid=6d0 op=createfilea start=168827052355 end=168827052924
40.06180954 ntfs_tracer: pid=6d0 op=WRITE start=168827053222 end=168827053392
40.06187820 [1744] apihijacking: pid=6d0 op=writefile start=168827053208 end=168827053429
```

The collected information will allow us to measure the performance penalty of the hDLP in the context of overall I/O processing time, and to focus on the areas that require improvements. We plan on profiling the I/O requests of the guest domains with different features of hDLP enabled to measure the overhead of individual components, and to present the detailed breakdown of the I/O operation execution time.

## 4.7 Security evaluation of Websense Endpoint DLP

Since SMW heavily relies on the security features provided by the Websense Endpoint DLP — and as such, can be viewed as a sort of competitor to our provenance tracking technology —- we also provided an assessment of its capabilities to prevent data leakage. To do so, we devised a protocol that tests components of a Websense Endpoint DLP solution with varying initial conditions to show the effectiveness of the tested solution. The goals of the exercise were to determine which policies could be circumvented at what privilege levels, and to monitor the responses of the Data Loss Prevention mechanism in terms of logging and blocking the attempted attacks.

The Websense Data Endpoint implements[5]:

- restrictions to copy and paste and printing operations (within applications)
- restrictions to file input and output
- restrictions to web and network traffic
- discovery of sensitive information using regular expressions

We found that the copy and paste, web and network policy enforcement was implemented in userspace using dynamically-linked libraries, enabling circumvention with a library redirection technique. Flaws in the implementation of the minifilter driver, which was used to implement other protections, allowed us to bypass the file copy restrictions. We found vulnerabilities that allowed for a **complete circumvention** of the Websense Endpoint DLP product for both privileged and unprivileged users.

Providing local administrator access to machines storing sensitive data is not desired due to security concerns, but providing better usability and offloading responsibilities to install and con- figure applications to users often outweighs the security concerns. The providers of DLP software are aware of the untrusted administrator scenario, and have included countermeasures intended to prevent un-skilled users from disabling the DLP. Unfortunately, the implementation flaws render the DLP software easily bypassable by a knowledgeable adversary. Additionally, the threat model assumed by the developers of Websense is currently unknown to us, and there are no published best practices on how to harden DLP deployments. We argue that even after fixing the implementation and design flaws the privileged user attack required to bypass driver-level enforcement still will be a valid concern.

[5] List based on information from the vendor:
http://www.websense.com/content/support/library/data/v753/install/endpoint_agent.aspx

## 4.8 Second Technology Preview

In the following sections we discuss the final performance results, system requirements, detailed system architecture, system deployment process and the self-guided demonstration. The project deliverable contains all the necessary components for a deployment of the hDLP data loss prevention system along with a sample guest virtual machine and sample configurations and tools provided for the purpose of self-guided demonstration.

Recall that in the earlier section, we discussed the performance impact of the file access tracking and enforcement. Then, we studied the overhead of the memory tracking and improvements to file access tracking. For more details refer to figures: 17 and 18. In the pilot studies we received positive feedback from users who did not notice performance impact on the systems protected using the hDLP platform, which was at odds with the CPUSpec benchmarks[3]. To get a better sense of user-perceived overhead, we decided to explore the use of another, more appropriate, benchmark. Specifically, we ran a set of tests from the PCMark 8 benchmark suite. The PCMark 8 suite is designed to replicate various types of user workloads (Home, Creative, Work, Storage, Applications). The Work workload replicates a typical office use ( spreadsheet, text editor, video chat and web page navigation), thus it was chosen for our evaluation. Unfortunately, Microsoft Windows 7 running under Xen/QEMU does not support the Video Chat benchmark, consequently we removed that benchmark from the test workload. We compared the median runtime of the benchmarks on the machine running the hDLP platform with the machine running unmodified Xen hypervisor.

Overall, the results (presented in Figure 19) show negligible performance impact. In fact, the observed difference (of 1%) is within the measurement error. We believe this benchmark better reflects our own experience using the system, and shows that the 2-8% performance impact observed using the synthetic benchmark CPUSpec 2006 was misleading. That is, for day-to-day real world usage, the perceived overhead should be barely noticeable.

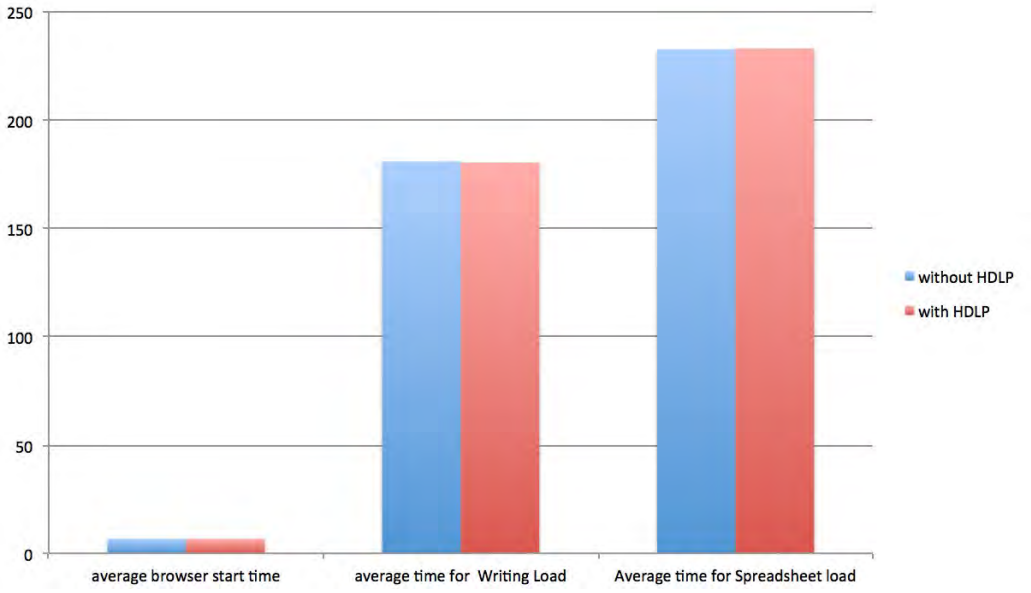


Figure 19: Results of the PCMark 8 Work workload

## 4.9 Capabilities of the reporting system.

To demonstrate the audit capabilities of the hDLP platform we built a web-based interface that provides access to the audit trail and detailed event reports. At a high-level, we provide both a data-centric and a user-centric view of the data. The data-centric (conceptualized in Figure 20) view allows one to view to examine the provenance trail, from the perspective of the sensitive data. By contrast, the user-centric view (shown pictorially in Figure 21) allows one to easily view all the actions taken by a particular user.

We presented the following scenario during the Presentations and Technology Demonstrations at the 2014 Cyber Security Division R&D Showcase and Technology Workshop: a researcher is granted access to the virtual data enclave provisioned with electronic medical records. She is allowed to access all the documents and create new content. Access to the files containing sensitive records is audited and new documents created using the data derived from the protected files are also protected. After her work session is completed the actions she has taken are audited.

An operator performing the audit is greeted with a dual pane page that provides the most important information at a glance as shown in Figure 22. This main page of the reporting interface we built provides a list of the most recent events in the monitored data enclave, navigation to data provenance visualization, a real-time view and detailed event reports.

The results of the data tracking between the documents in the protected enclave are presented in a visual way that allows one to track the provenance of data. The interface presented in Figure23 enables identification of data sources and data sinks.

The sensitive objects are represented as labeled nodes of the graph while the edges connecting the nodes denote specific events that moved data between locations. In this report, we observe two initially accessed documents being sources of the sensitive data copied to intermediate locations and finally saved to a final destination.

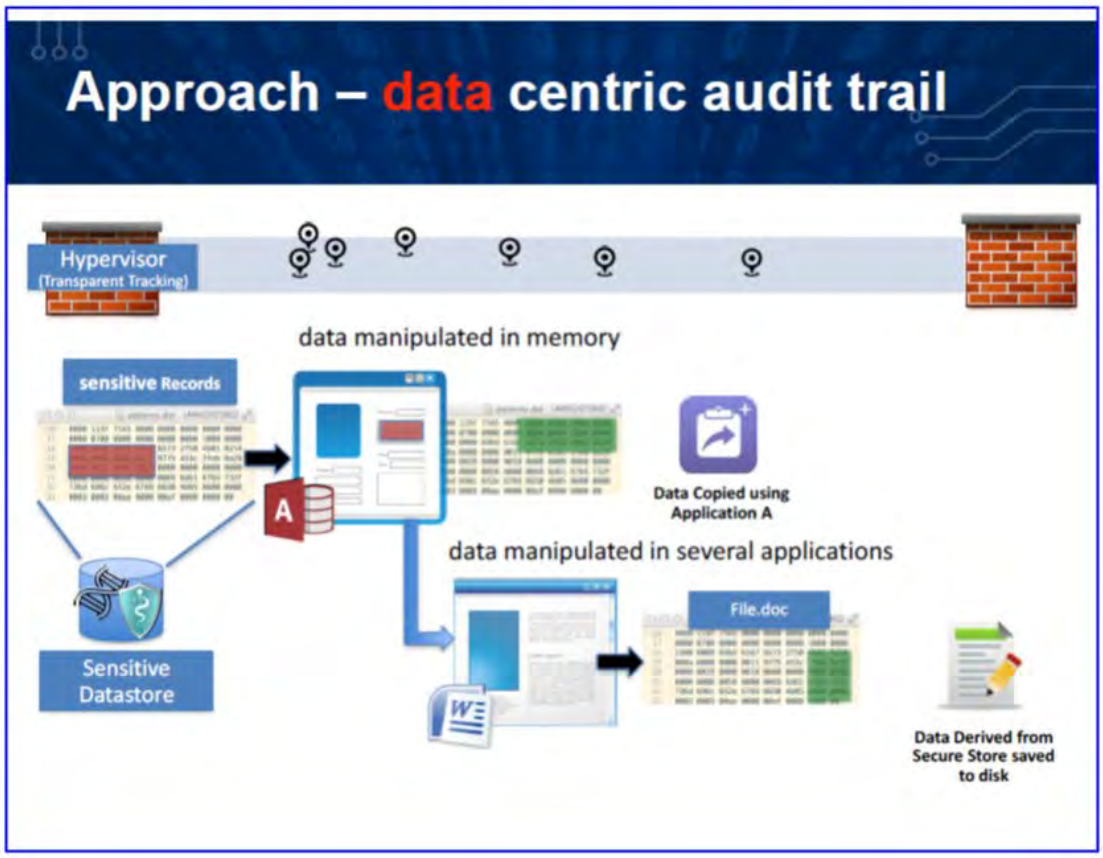


Figure 20: Conceptualization of the data-centric view of the audit log

Selecting an edge from the graph navigates to a view with detailed event information. Figure 24 depicts the interface that provides access to the details of an event. The forensic analyst using this view is presented with details about protected files and the applications that accessed sensitive data. Additionally, the interface displays the size and contents of the tracked memory that was lifted from the source and written to the destination file. In this example we observe a process named `tc.exe` copying a data buffer from an electronic medical record file named: `"monicalattemer.txt"` to a file named `"leaked doc2.txt"`. The shown data buffer contains information about the access, confirming that the data moved to a new file came from a protected document.

Using the reporting interface an operator can navigate to the main window as well as to the real-time event viewer. The real-time event viewer, presented in Figure 25, allows an operator to observe the activity of the system, and offers a convenient way to access the details of events.

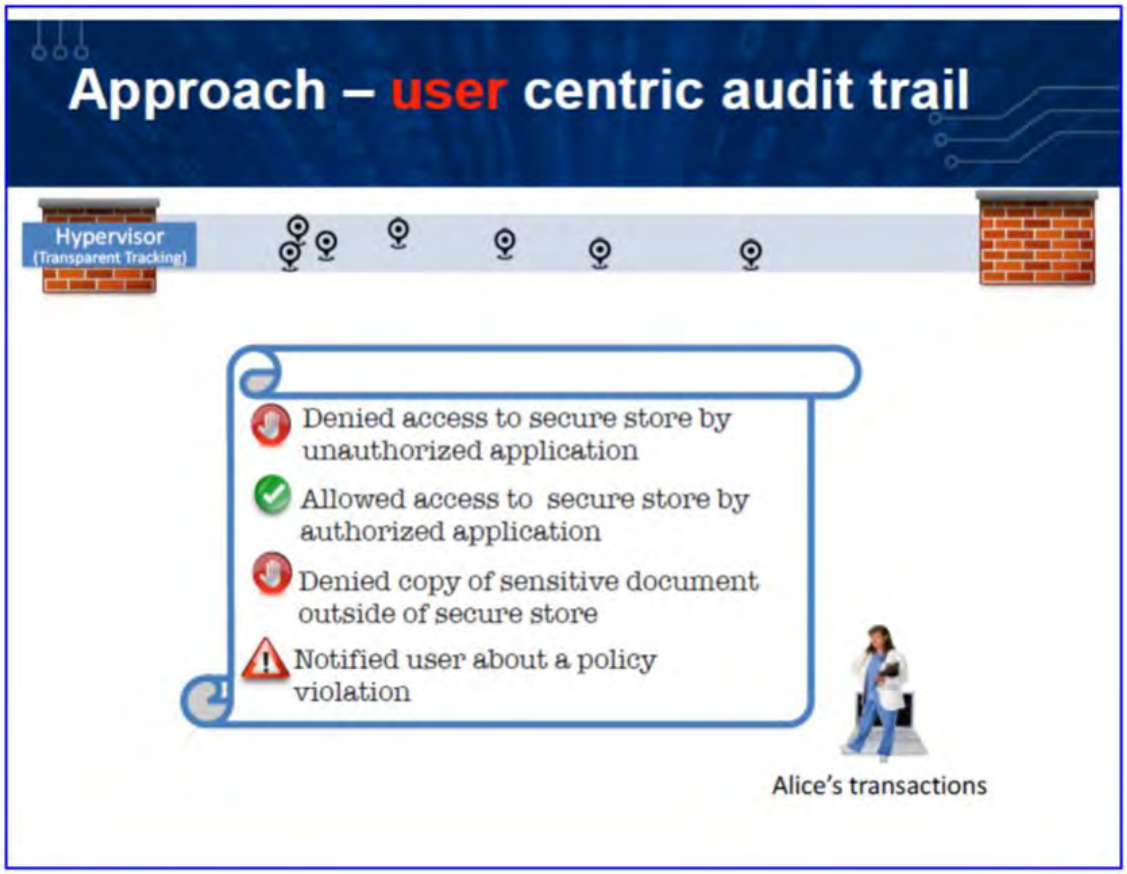


Figure 21: Conceptualization of the user-centric view of the audit log

## 4.10 Review of Hardware and Software Requirements

The deployment of the hDLP platform requires the following prerequisites.

1. A *physical* desktop or server machine with at least 16 GB of RAM, 50 GB disk, and a 64- bit Intel Xeon, Core, or Pentium line processor with 4+ physical cores and *VT-x* (hardware virtualization) support[6]. Most recent machines meet these requirements. Known compatible[7] hardware include:

    - Dell Optiplex 990 Desktop (Intel Core i7-2600, 16GB RAM)
    - MacBook Pro 10,1 Retina (2013) (Intel Core i7, 16GB RAM)

2. A wired Ethernet connection. Xen requires the wired Ethernet connection to bridge the network access to guest virtual machines.
3. A *Ubuntu Desktop 14.04.1 (64-bit) installation*.[8]
4. A valid *Windows 7 Professional* product key.

[6] We *do not* support installing within a virtualized environment, e.g. VMWare.
[7] We *do not* support AMD processors due to the lack of support for specific virtualization extensions. See §7
[8] We provide an installation DVD for Ubuntu as the part of the final package

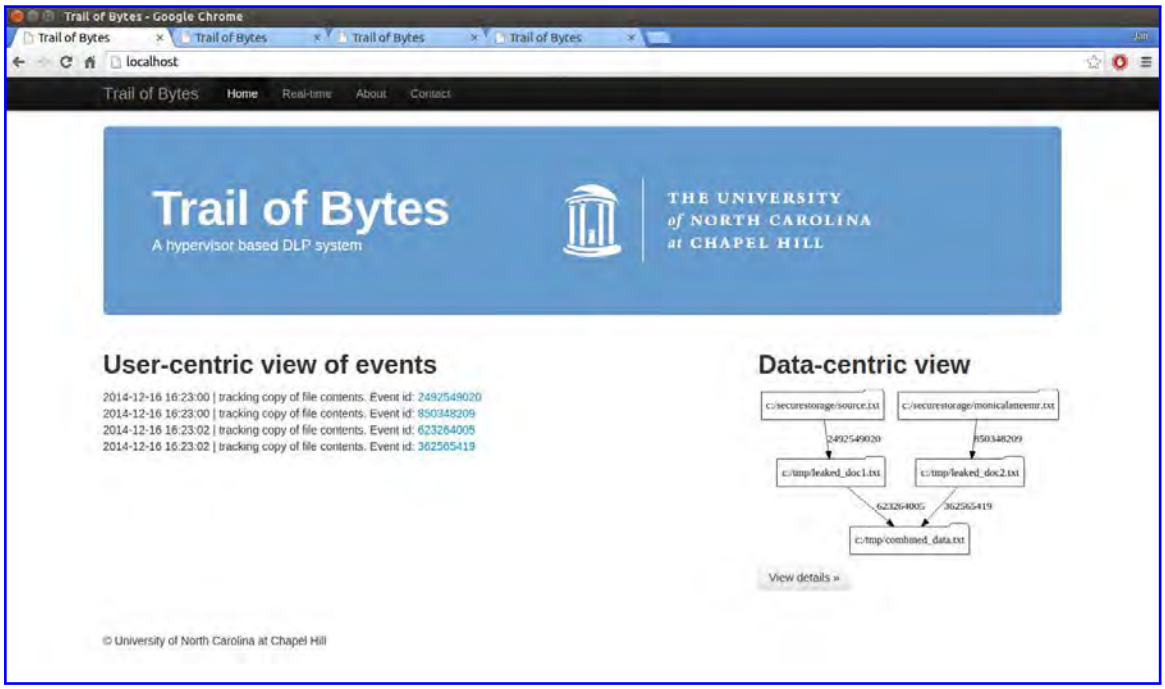


Figure 22: Implemented interface showing the user and data centric event views.

## 4.11 hDLP platform components description

The hDLP system consists of following components presented in Figure 26. We briefly elaborate on each component below.

❶ The core data loss prevention system implemented as an extension to the Xen hypervisor. The hDLP system is provided as a patch for the current version of the 4.4 branch of Xen (4.4.2 as of March 31st, 2015). This component is located in the directory "hDLP core" on the hDLP Platform Preview DVD.

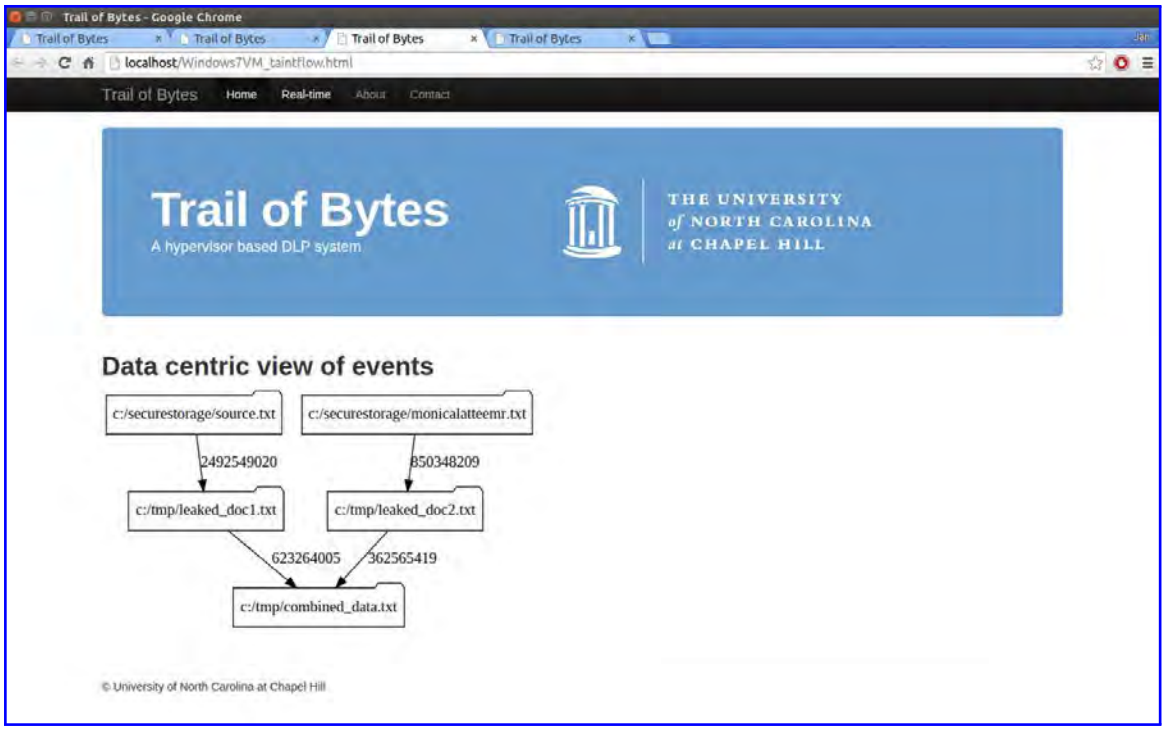


Figure 23: Implemented visualization of the data flows between files.

❷ The DLP configuration utility running in the management domain. This utility allows for management of the hypervisor DLP module on per domain basis. Along with the tool we provide a sample configuration file used to configure the data loss prevention for the Windows 7 guest virtual machine. This component is located in the directory “hDLP core” on the hDLP Platform Preview DVD.

❸ The data storage module implemented as a kernel module running in the management domain. This module provides a mechanism for data transfer between the hypervisor and the monitoring tools running in userspace. For a detailed module description please refer to Section 9.2.1 and the documentation included in the final deliverable package. This component is located in the directory “KBufDevice” on the hDLP Platform Preview DVD.

❹ The monitoring tool implemented as a python script running in the management domain. This tool monitors the output of the hypervisor logs, orchestrates the extraction of an event data from the hypervisor and generates detailed audit trail entries. For a detailed module description please refer to Section 9.2.1 and the documentation included in the final deliverable package. This component is located in the directory “hDLP monitor” on the hDLP Platform Preview DVD.

❺ The storage device access tool implemented in GoLang running in the management domain. This tool is used to extract files from the guest virtual machine storage devices. Extracted files are attached to the audit trail entries allowing the analyst to observe how the tracked data has been used to create a new document. For a detailed module description please refer to Section 9.2.1 and the documentation included in the final deliverable package. This component is located in the directory “BlockFinder” on the hDLP Platform Preview DVD.

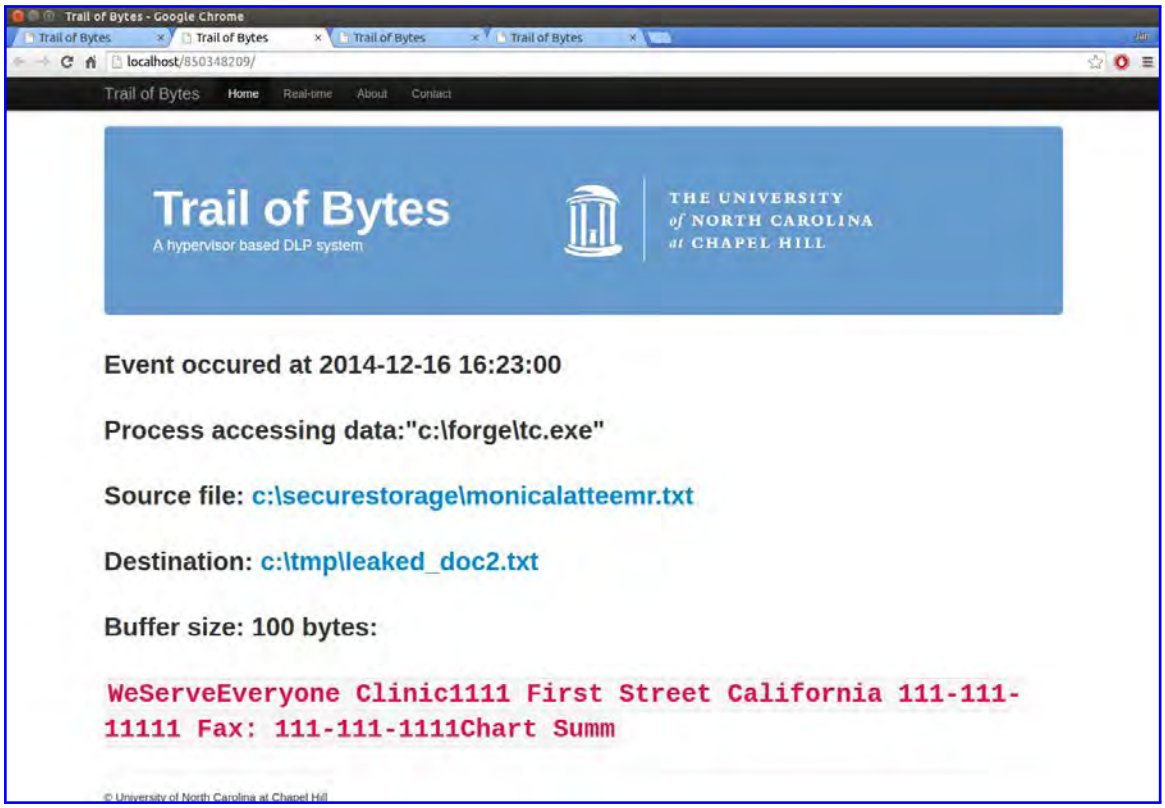


Figure 24: Implemented event forensic information viewer.

❻ The Web-based frontend backed by the nginx web server provides access to the audit events and real-time event monitoring interface. For a detailed module description please refer to Section

❼ Guest Virtual Machine with pre-installed Microsoft Windows 7. We provide a configured deployment with tools for for self guided evaluation of the hDLP Platform. This component is located in the directory “guest VM” on the hDLP Platform Preview DVD.

The final deliverable DVD additionally contains:

1. Tools used to demonstrate the vulnerabilities in the Websense DLP (see section §6.1.3) located in the directory “WebsenseDLP tools”.
2. Full report of the security evaluation of the Websense DLP (see section §6.1.3) located in the directory “Documentation”.

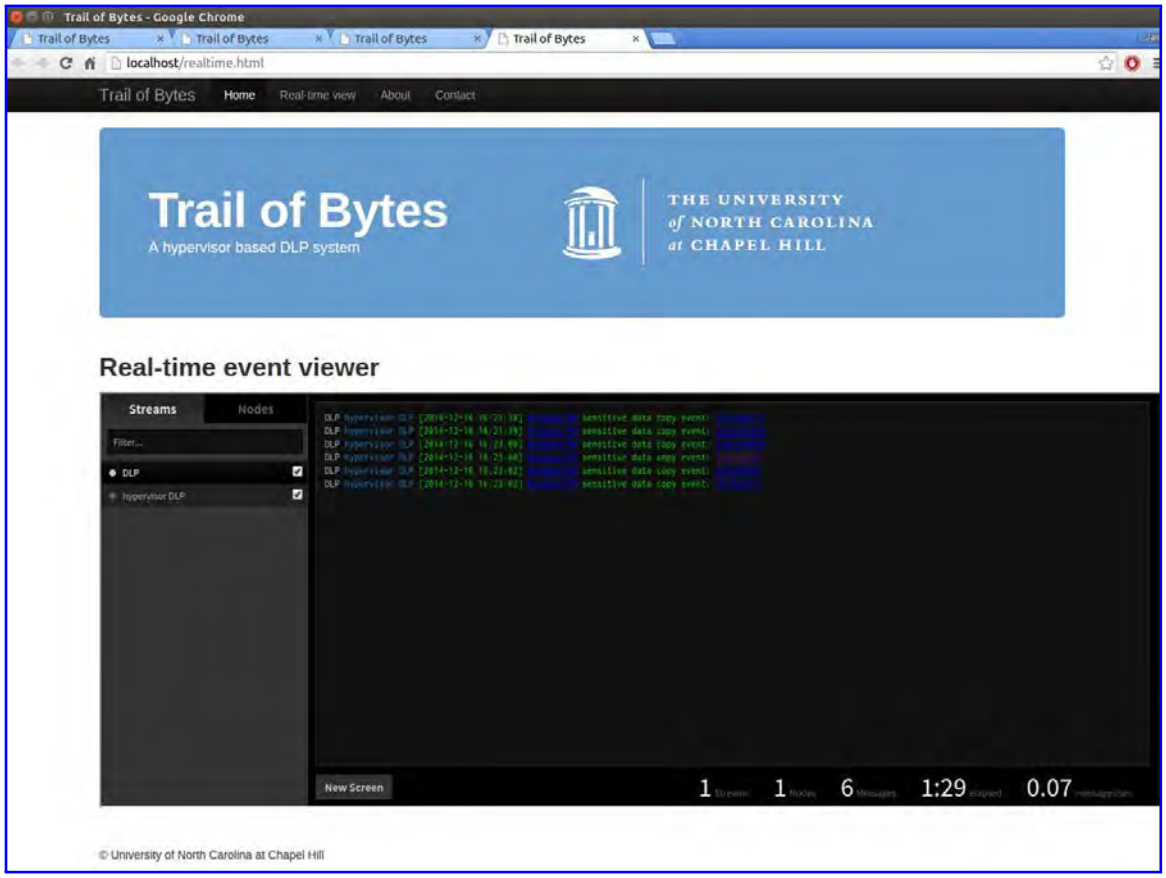


Figure 25: Implemented real time event viewer.

3. A tool for tracing file system access developed by the University of Michigan (see section §8.2.2) located in the directory "Trace tools".

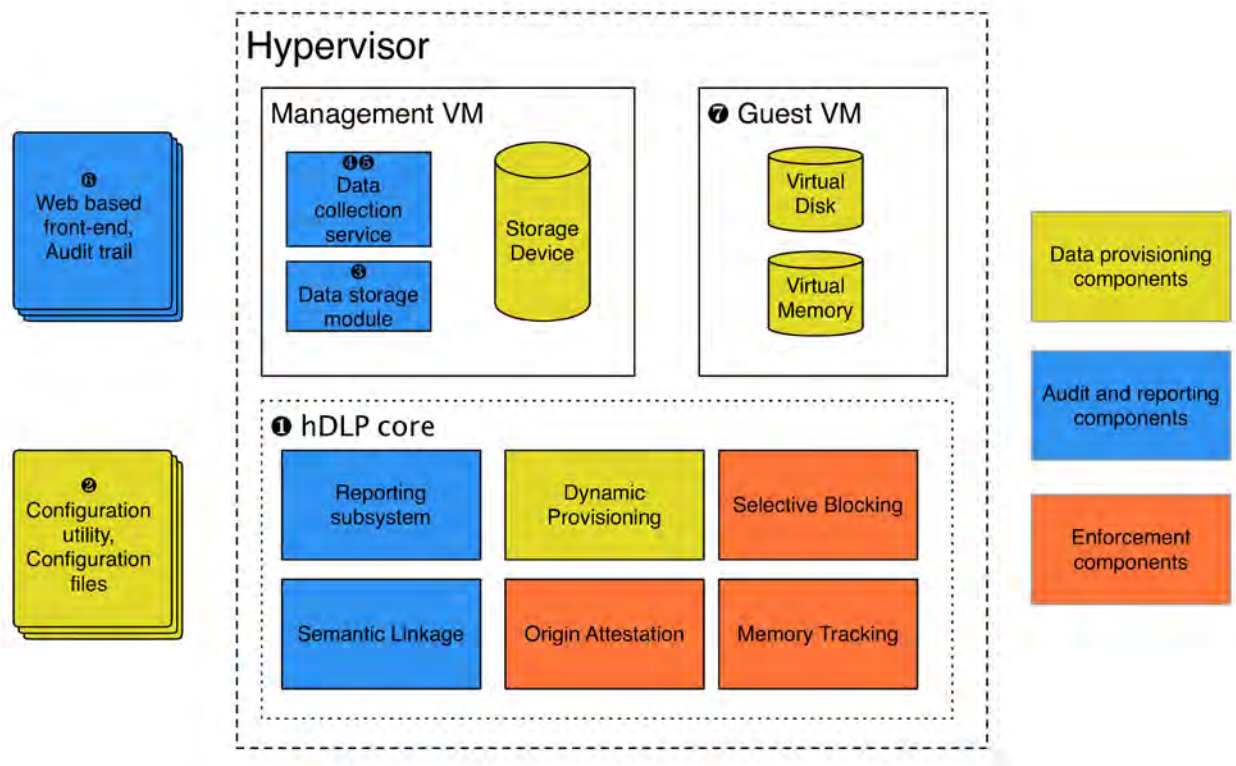


Figure 26: Annotated architecture of the hDLP platform

## 4.12 Deploying the hDLP platform

1. Install *Ubuntu Desktop 14.04.1 (64-bit)*. Step-by-step instructions are available on the Ubuntu community website: `http://www.ubuntu.com/download/desktop/install-ubuntu-desktop`.

   (a) Boot from the *Ubuntu Desktop 14.04.1 (64-bit) installation* DVD, select "*Install Ubuntu*".

   (b) Select "*Download updates while installing*" and continue.

   (c) To install to disk, select "*Erase disk and install Ubuntu*".

   (d) Follow on-screen instructions for timezone and keyboard setup.

   (e) Create a user account and remember the password for later use.

   (f) After installation completes, remove the installation media and reboot the computer.

2. Install Ubuntu system updates.

   (a) Login with the account created.

   (b) Install system updates by clicking on the bottom-left icon in the left-hand taskbar called "*Software Updater*". It may take a few minutes for this icon to appear and you must be connected to the Internet.

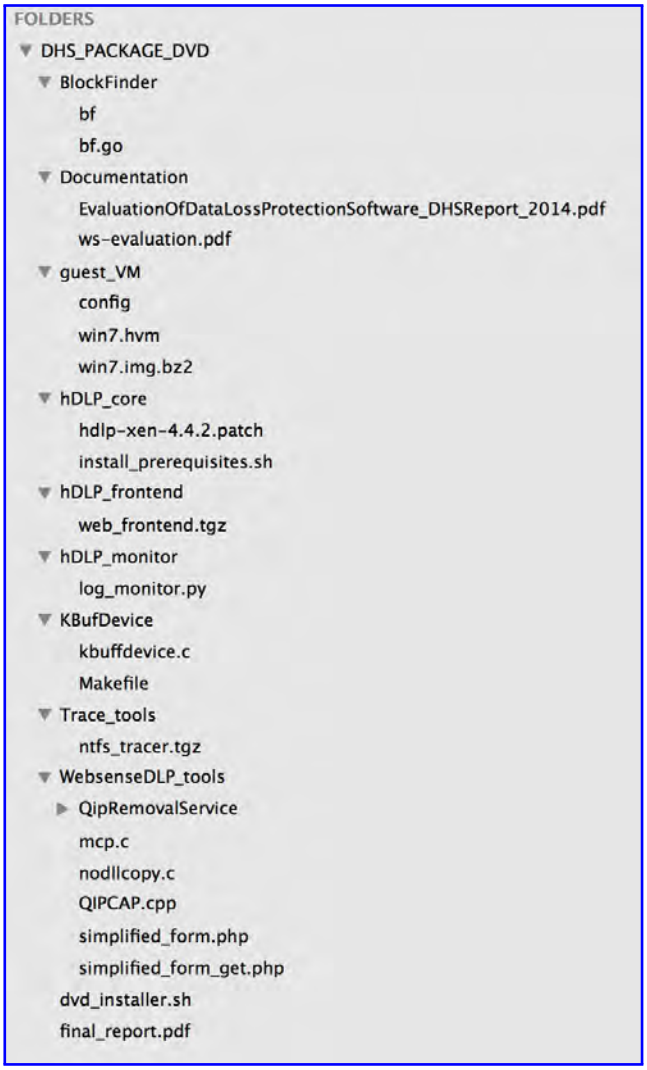


Figure 27: Contents of the hDLP platform preview DVD

(c) Click "*Install Now*" and authenticate with your account password.

(d) Click "*Restart Now...*" when finished.

3. Install the *hDLP Platform Preview*.

**Only start hDLP installation after system updates complete, they can not be run at the same time.*

(a) After logging in, insert the *hDLP Platform Preview installation* DVD.

(b) If not automatically prompted, click the DVD icon in the task bar, then *Run Software* in the file manager.

(c) At the prompt "*'hDLP' contains software intended to be run automatically. Would you like to run it?*", click *Run*.


Figure 28: Desktop of the system with hDLP platform preview

(d) The terminal opens, enter the password you created during installation of Ubuntu. *Ensure no warnings arise regarding minimum requirements.* This takes a few minutes and you may be prompted to enter your password again.

4. Restart the computer. *hDLP Platform Preview* installation will be completed after the reboot
5. Activate Windows 7 [9]
   - On the desktop find the startGuest icon. Double click it to start the guest virtual machine and the hDLP platform. If prompted press run in terminal.
   - A terminal window will open along with a VNC viewer of the protected Windows 7.
   - After it boots, select "*Type your product key*", type your key, then click *Next*.

     *note:* if Windows boots straight to the desktop, simply click Start Menu → *Activate Windows* to type your key.

- Close the "*Activation was successful*" dialog.

[9] Without a Windows product key you can still evaluate for 30 days

## 4.13 Self guided demonstration

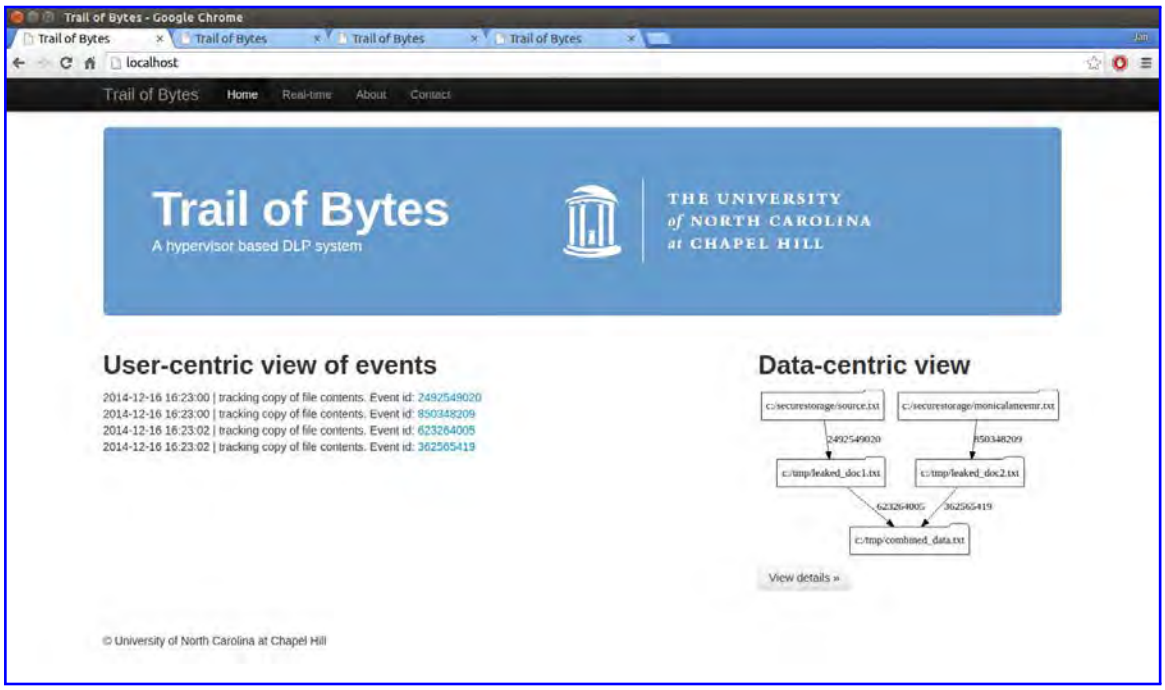


Figure 29: The frontend interface

After installing the hDLP Platform Preview, one can run the protected guest virtual machine.

1. Start the guest virtual machine by double clicking the startGuest icon (Figure 28), on the desktop. Wait until the Windows 7 guest Virtual machine has started.
2. In the host Ubuntu open the *Firefox* web browser from the task bar and browse to "localhost".
3. You should see the hDLP monitoring frontend (Figure 29), a web application running locally.
4. The front end will display information about previous events.[8][10]
5. Switch to Windows 7 guest. On the desktop you will find sample application used to demonstrate the memory copy tracking. Double click to execute the the test tracking program (Figure 30).
6. Switch back to host Ubuntu, switch to the *Firefox* web browser for the event log and detailed information.

[10] On the first start it will present no information.

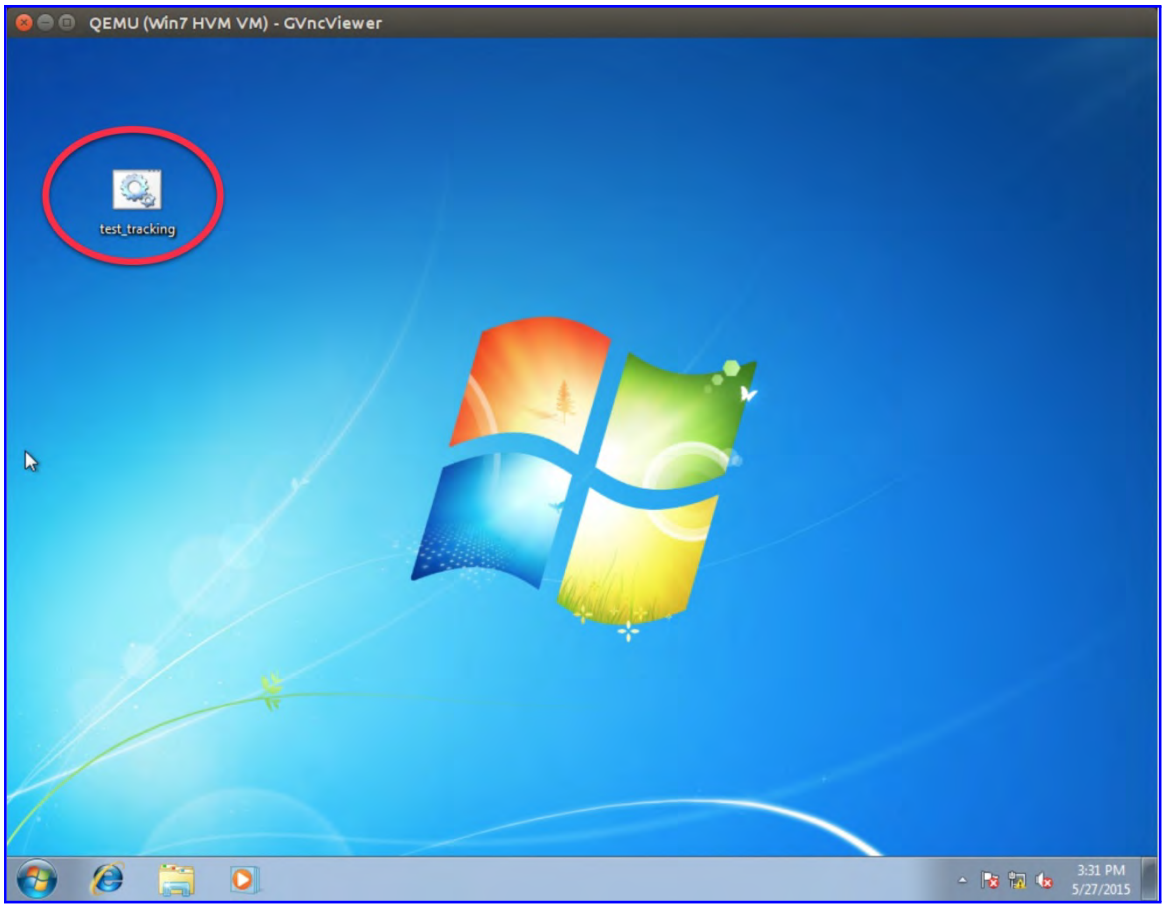


Figure 30: Desktop of the test Windows 7 Virtual Machine for hDLP platform preview

# 5 Conclusions

The hDLP platform has proven to be an efficient and effective solution for the problem of preventing data leaks. The delivered solution provides two different levels of protection granularity:

- file level protection that confines the files to predefined secure storage devices (see Task 4.3 in §5.1)
- byte level protection that allows for tracking the movement of the memory contents with (see Task 4.5 in §6.1) byte level precision.

The combination of the two approaches places the capabilities of the hDLP above of the state of the art commercial Data Loss Prevention systems.

The delivered solution have limitations arising from the architecture design and implementation choices. Our platform currently supports: single processor virtual machines running a 32-bit version of Windows 7. In more detail:

- Handling multi-processor VMs would require additional verification of the state of the virtual machine. Tracking and enforcement mechanisms of the hDLP must guarantee that they will not corrupt the state of the protected machine during the manipulation of the guest state.

- Providing support for both 32-bit and 64-bit virtual machines requires additional logic to process 64-bit instructions in memory tracking, changes in the handling of the system call mechanism (different instructions are used in 32-bit and 64-bit modes of operation) and modifications in the mechanisms of data extraction from the guest virtual machine.

- Enabling the hDLP platform in different versions of Microsoft Windows requires additions to the system call handling mechanism (system call numbers differ between releases of Microsoft Windows) and modifications in the mechanisms of data extraction from the guest virtual machine (opaque data structures in the Windows kernel may be different between versions).

A continuation of work on the hDLP should begin with addressing the limitations listed above. Addressing those issues would allow the hDLP to cover much larger share of the workstation configurations. We suggest making improvements to memory tracking mechanism: the provided implementation handles several common scenarios of the in-memory data copies, but it fails to cover all possibilities, including implicit data flows. Finally, it might be useful to explore alternative approaches for the Data Loss Prevention solution. One interesting direction would be a hybrid between the hDLP and in-guest DLP systems. Instead of relying on modifications to a complex hypervisor like Xen (hDLP approach), one could implement the data loss prevention functionality using a thin hypervisor. A thin hypervisor would depend much more on the guest operating system API, and would require less code running in the hypervisor. This approach would therefore inherit the best of both approaches: tighter integration with the guest OS kernel provides better interfaces to track user actions, and running as a hypervisor provides isolation from potentially a malicious guest and allows for fine-grained memory tracking.

## 5.1 Technical Points of Contact

The technical points of contact for this contract are Dr. Fabian Monrose (fabian@cs.unc.edu) and Jan Werner (jjwerner@cs.unc.edu). Any inquires on technical matters contained herein should be directed to those email addresses.

# 6 Privacy Threshold Analysis Statement

We remind the reader that we built a tool for researchers to protect private information that already exists in their datasets. At no point did we collect personally identifiable information (PII) about the researchers, nor did we collect any datasets that contain PII that researchers use. That being said, our tool may eventually be used by a researcher who wants to protect a dataset they have in order to guage the effectiveness of our tool. In such cases, even though we did not store or process the researcher's dataset ourselves, we ensured that the researchers we used during the pilots and performance evaluations collected their datasets ethically.

# 7 References


[1] F. Buchholz and E. Spafford. On the role of file system metadata in digital forensics. *Digital Investigation*, 1(4):298–309, 2004.

[2] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualiza- tion extensions. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 51–62. ACM, 2008.

[3] J. L. Henning. SPEC CPU2006 Benchmark Descriptions. *ACM SIGARCH Computer Archi- tecture News*, 35(1), 2007.

[4] M. Shoffner, P. Owen, J. Mostafa, B. Lamm, X. Wang, C. Schmitt, and S. Ahalt. The secure medical research workspace: an it infrastructure to enable secure research on clinical data. *Clinical and Translation Science*, 6(3):222–225, April 2013.

[5] G.-R. Uh, R. Cohn, B. Yadavalli, R. Peri, and R. Ayyagari. Analyzing dynamic binary instru- mentation overhead. In *WBIA Workshop at ASPLOS*, 2006.

[6] X. Wang, J. Zang, Z. Wang, Y. Luo, and X. Li. Selective hardware/software memory virtual- ization. *ACM SIGPLAN Notices*, 46(7):217–226, 2011.

# Appendix A: Data Leakage Prevention EndPoint Protection Software: Evaluation for Use in Secure Workspaces

Michael Shoffner, Xiaoshu Wang, Fan Jiang, Charles Schmitt

University of North Carolina at Chapel Hill

## Executive Summary

A secured IT environment is required to allow work using sensitive electronic data to occur while minimizing risk of intentional or unintentional loss of the sensitive data. A key technology for providing a secured environment is Data Leakage Protection (DLP) software. DLP software monitors and potentially blocks the flow of data from a secured environment to an unsecured environment based on attributes of the data itself. Commercial DLP products focus on three mechanisms; discovery of the location of sensitive data (Discovery DLP); monitoring and blocking of data as it moves through a computer network (Network DLP); and monitoring and blocking of data as it is copied off of computing devices (EndPoint DLP).

In support of biomedical research, UNC has developed an IT environment termed the Secure Medical Workspace (SMW) that relies upon commercial EndPoint DLP to prevent loss of sensitive patient records. An initial evaluation of commercial EndPoint DLP offerings was conducted in 2010---2011 for use in the SMW. This paper provides a re---evaluation of commercial EndPoint DLP offerings with a focus on weaknesses in commercial offerings that were identified at the time of the initial review.

The primary result of this re---evaluation is that we have seen a maturation of the commercial DLP offerings. The maturation of the market is reflected in: greater bundling of DLP solutions with other security technologies (e.g, network monitoring and firewalls, anti---virus) into enterprise solutions; modest decreases in pricing; improvements in approaches for identifying a data set as containing sensitive data (in particular approaches for fingerprinting of files and tagging/tracking of files); increases in the number of features offered by even the low cost DLP offerings; and improvements in overall ease--- of---use. However, there are still several weaknesses of concern when one considers DLP technology as a key component of a secured environment, principally: EndPoint DLP is rarely offered for non---Microsoft Windows environments; EndPoint DLP technology remains exposed to security threats which could disable the DLP technology; tagging/tracking of sensitive data, a powerful technique for identifying sensitive data, is implemented in only a few vendor solutions and tends to be implemented through moderately secure approaches (e.g., use of file system metadata).

To summarize, the DLP Endpoint market continues to offer powerful technologies for reducing the risk of data loss from a secured environment. However, in the face of extremely costly financial losses that can occur from data loss, the commercial solutions continue to have several concerning deficiencies.

## Introduction

In 2011, RENCI in collaboration with the UNC Healthcare System and the UNC Information Technology Services (ITS) group, developed a solution to allow clinicians, IT staff, and researchers to securely work with patient electronic medical records that include protected health information (PHI). The solution, termed the Secure Medical Workspace (SMW), is now in production use at UNC.

The SMW solution relies heavily on the use of Data Leakage Prevention (DLP) technologies that can audit and block (or challenge) the removal of PHI from an IT environment. More specifically, the SMW relies on EndPoint DLP technology; a variant of DLP technology that runs on the computer from which sensitive data is accessed. An initial review of Endpoint DLP technologies was conducted in 2010---2011 by RENCI to identify commercial solutions capable of meeting SMW requirements. From the review, the WebSense commercial technology was adopted for use in the SMW. However, multiple deficiencies in Endpoint DLP offerings from all commercial vendors were recognized at the time of the initial review. This document is an updated review of Endpoint DLP commercial offerings with a focus on where the state of Endpoint DLP commercial technology is in regards to the existing list of requirements for the SMW environment.

## Background on DLP

To secure an IT workspace requires imposing measures that can be used to guard against potential security breach targeted toward the protected workspace. Here, the word "workspace" is used in a very general sense. It can refer to either a physical, a virtual environment, or a cooperate network where data is hosted and/or computed.

Depending on the direction of information traffic to be guarded, security software is divided into two groups (See Figure 1). Software that guards against in---bound traffic is intrusion protection system, which can be further divided into Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). The former system detects, whereas the latter responds to, security threats. On the other hand, software system that guards against the security breach from out---bound traffic is called Data Loss/Leak Prevention (DLP) software, which is the main focus of this review.

Although similar in nature, intrusion prevention system and data loss prevention system differ in purposes. Intrusion prevention system aims at preventing an outside attack, such as denial of service, and the penetration of active content with malicious intent, such as virus and malware etc. DLP system, on the other hand, is aimed at protecting sensitive data from leaking out an organization. The protected content is mainly of two kinds. The first is private identifiable information (PII) and the second is intellectual property (IP). PII is more of private information that secretive. The concern of protecting PII is often aimed at complying with governmental or industrial compliance audit. IP, on the other hand, is more of secretive in nature. Compared to PII, which often has a straightforward definition, such as credit card number and patient's medical billing code, etc., IP's definition can be very broad. Protecting IP thus often requires more sophisticated detection algorithm and more complex management capability from a DLP product.

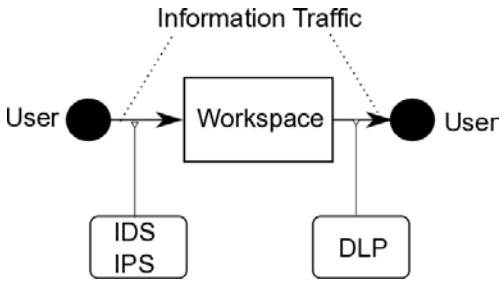


Figure 1: Security Software Classification

## Main Features of DLP Software

Two major features distinguish DLP software from other security software. The first feature is that DLP's capabilities are always based on a set of central policies. In addition, these policies can be controlled/configured by appropriate personal with regard to what kinds of data needs to be identified, what kind of activities need to be monitored, and how their usage should be protected. This controllable policy---based feature distinguishes DLP product from other products, such as an enterprise firewall, which security policy, once set, rarely needs to be controlled by an IT administrator.

The second feature of DLP software is its content---awareness. DLP software must have the capability to inspect the content of a data, often at runtime, in order to identify, classify, and subsequently react in accordance to a predefined policy. This content---awareness distinguishes DLP product from the traditional device control technologies, which relies on data's context, i.e., location and users, which are indirectly regulated by an external access control mechanism. Compared to context---based protection, DLP technologies enables data protection to be carried out on a much finer granularity.

Nevertheless, DLP should not be considered to be an antagonist to, or a replacement of, context---aware technology. In fact, these two technologies compliment with each other because whether the manipulation of a sensitive data violates a security policy cannot be solely determined by the nature of data. The enforcement of a DLP policy, in fact, can only be meaningfully applied when it leverages data's context, i.e., who is using it and where the data comes from, and where it is sent to.

## Market Scope

Comprehensive protection of a workspace requires security measures to be enforced at multiple levels. Data's content inspection is just one of the many facets of the entire security requirement. The purpose of focusing on the above two features, i.e., content---awareness and policy---based action, is to narrow the scope of the market landscape. Thus, if a software product does not make its entry into in this review, it does not imply that the product is not useful in terms of securing a workspace. Take IPSwitch's product (MOVEit DMZ) as an example. As MOVEit DMZ allows customized policy to be set upon the network transfer of a particular file, it can obviously be used to protect a designated workspace. However, because MOVEit DMZ does not perform content inspection, it is therefore not considered to be a DLP product. Gartner's group, for instance, considers IPSwitch a leader in the market for managed file transfer (MFT) (See Figure 2) and does not mention it on the magic quadrant for DLP (See Figures 3 and 4).

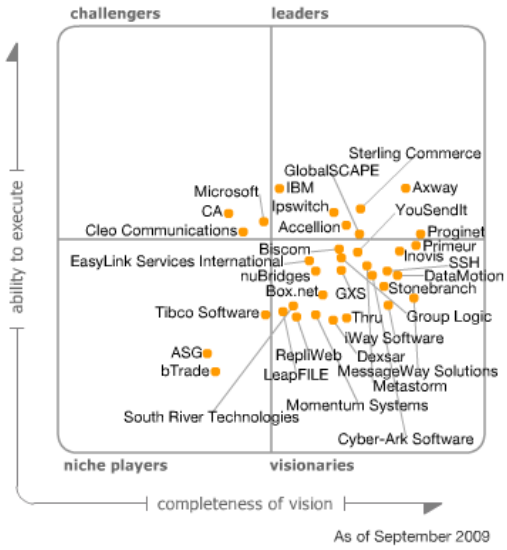


**Figure 2: Magic Quadrant for Managed File Transfer**

Another prominent omission in the review is the VMSafe technology developed by VMWare. The reason for its omission is due to the different levels of protection. DLP offers protection at data--- or application--- level. VMSafe, on the other hand, offers protection at platform---level. Obviously, platform protection can only help data--- or application---protection. As an operating system's security API, VMSafe can be an extremely useful mechanism for engaging a DLP product. However, such a difference makes it difficult to compare VMSafe with other DLP products. Hence, we have omitted such technologies in this review.

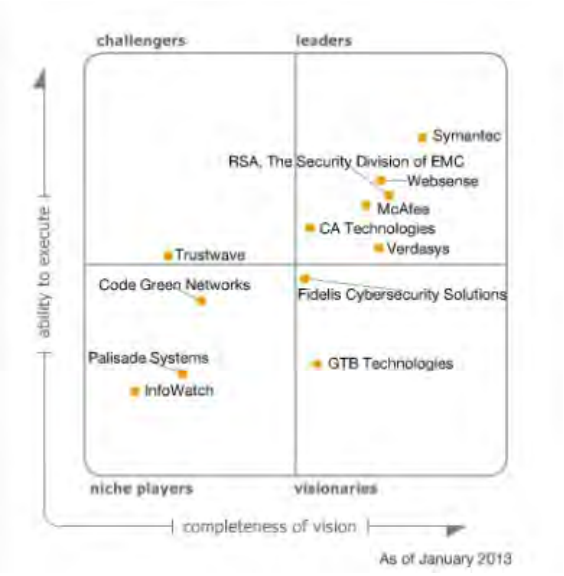


**Figure 3: Magic Quadrant for Content---Aware Data Loss Prevention, 2013**

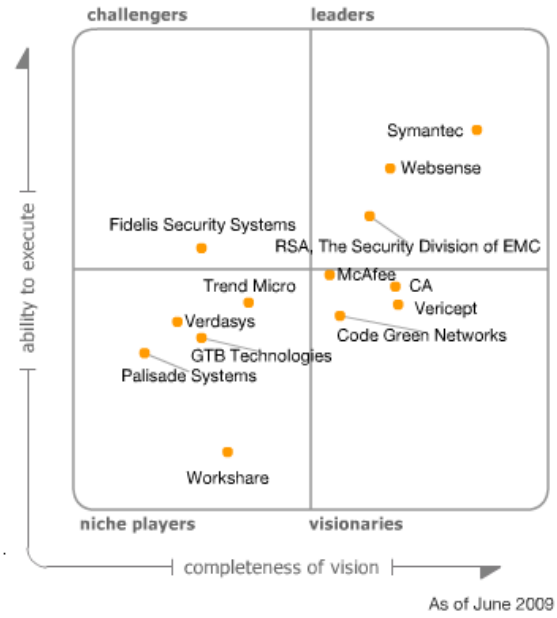


**Figure 4: Magic Quadrant for Content---Aware Data Loss Prevention, 2009**

Naturally, there could be some other software that we have omitted due to the rapid evolution in security solutions being created by vendors and due to vendors marketing products under different terminology. As the DLP software market has become more competitive in recent years, some vendors have started to market their DLP capability as an added feature to their products that is used for a different purpose. For instance, Clearswift's products are both policy---based and content---aware. However, the company brands its product as a content---aware secure web/email gateway, as opposed to a DLP product. Because of this branding strategy, it is very likely that we could have missed some relevant products.

# General Architecture of DLP Software

## DLP Channels

A DLP product can monitor, identify and protect the data in three primary channels: network, endpoint, and storage. Network DLP inspects data in motion, i.e., those data flowing on network traffic, such as HTTP, FTP, IM, or Email etc. Endpoint DLP monitors the data in use, e.g., the data to be copied to a portable or network storage devices, or the data that is sent to a printer. Discovery DLP, on the other hand, inspects data at rest, such as files and databases etc. The purpose is to identify where the sensitive data is located and whether a data should reside at certain location.

The above three channels of DLP products were traditionally independent. However, more and more vendors have started to consolidate three---products into a single suite and sell them as an enterprise DLP product. Although generally more expensive, enterprise DLP is advantageous in that it enables an organization to apply a set of centralized policies across all aspects of data movement within an organization. Nevertheless, most vendors continue to sell their DLP products in single---channel setting regardless of whether an enterprise solution is available. In addition, small vendors, who are seeking a niche in the security market, often blend their DLP capability as an added feature to their non---DLP related products, such as email, intrusion detection, and identity and

access management (IAM).

## Network DLP

All network DLP contain a passive network monitor, deployed near a SPAN port (or a similar tap) but within cooperate firewall (See Figure 5). The network monitor is usually a dedicated server with DLP software installed. For single---channel product, the server is also often used to manage the workflow, monitoring and reporting functionalities. For enterprise DLP, the latter functionalities are often offloaded to a different server for the purpose of a centralized policy control and hierarchical deployment.

All network DLP eventually need to filter and block network traffic. Such functionality can be done with either a bridge or a proxy. A bridge is simply a computer with two network cards on each side and it inspects the traffic content in the middle. If the traffic passes the content inspection, it is let go. Otherwise, it will be dropped. Technically, a bridge network---DLP inspects data at packet level, making it protocol antagonist. Although this may seem to be an advantage as all network traffic, regardless of the transportation protocol, can be regulated, bridge---network DLP is less effective in terms of data loss protection. Partial information, delivered in packet, could have been already leaked out before the inspection engine realized the problem and takes action. In addition, the inability to understand a protocol makes it unlikely to respond to a policy violation in appropriate fashion. The most that a bridge network---DLP can do is to drop an established TCP connection and logs the policy violation. However, it cannot meaningfully notify the offender, who may or may not have a malicious intent.

Almost all network---DLP products, therefore, use proxy---based technique . Messages will be queued up at proxy, allowing DLP software to perform thorough inspection of its content, before making a decision. Since proxy---based network DLP understands each protocol, it can offer not only better protection but also more meaningful remedial action. For instance, if DLP protection detects that a user is accessing a sensitive material through HTTP, it can return an HTTP response, containing a warning message or a disclaimer form, to the user instead of simply denying the access. Furthermore, proxy---based approach allows DLP product to be integrated with existing web gateway product, typically through iCAP protocol, allowing DLP vendors to avoid the difficulties of building a dedicated hardware for inline analysis. However, the down side of proxy---based techniques are that a delay is introduced on all traffic leaving a host computer, regardless of any existing knowledge about the data.

It is worth noting that Email DLP is sometimes offered as a separate DLP product, especially from relatively small vendors. The reason is that email data is first stored and later forwarded. Hence, it differs from most real---time synchronous network traffic. As this store---and---forward model is by nature a proxy model, Email DLP is relatively easy to implement. In addition, the storage model also gives DLP more capabilities in terms of response. Unlike real---time traffic, an email message can be quarantined or encrypted along with a notification to the sender for the potential policy violation or, in the case of a false positive, an instruction on how to decrypt and resend the message.

## Discovery DLP

Discovery DLP scans sensitive content from (1) network endpoint, i.e., workstations and laptops (2) network storage, such as file servers, SAN (Storage Area Network), and NAS (Network Attached Storage)

(3) application servers, such as email servers, document management systems, and databases.

A discovery DLP will be either server---based or agent based. A server---based discovery DLP uses a file--- sharing or application protocol to perform data scan remotely. An agent---based discovery DLP installs the DLP---product, either temporarily or permanently, on the system to be scanned. Typically, an agent---based discovery DLP is used as an added feature to an endpoint DLP.

Unlike what its name suggests, a discovery---DLP does more than passive scanning data repositories. Once a policy violation is discovered, discovery---DLP can take a variety of actions, such as to alert/report, to notify the user, to quarantine/encrypt, or to simply remove/delete the sensitive data. Discovery DLP can also be used to fingerprint data sets and the fingerprints can be used to inform Network and Endpoint DLP mechanisms.

## Endpoint DLP

Early endpoint DLP products were focused on monitoring and enforcement within the file system or network stack so that it can prevent sensitive data from being copied to a portable storage devices. Recent DLP products have started to incorporate their functionality at the system/kernel level. By plugging directly into to the operating system kernel, a DLP product can monitor more detailed user

activities, such as cutting and pasting sensitive content, and preventing users from using unauthorized method to encrypt sensitive information. Many DLP---endpoint products, such as CA and Websense, can prevent a user from printing materials containing sensitive information .

## Content Inspection Engine

At the core of all DLP product is a content---inspection engine. In essence, a content---inspection engine is a file---cracker that uses various techniques to read the content of a file, an embedded message, or a byte--- stream, etc. Most DLP has their own proprietary content inspection engine. Some combine its own engine with third party search engine, such as Autonomy. Most engines can identify standard encryption and use that as a contextual rule to block/quarantine content. Although proprietary in details, content--- detection engines all tend to use the following techniques.

**Rules---Based/Regular Expressions:** This is the most common analysis technique and used in early DLP product. The content's text will be analyzed for matching specific rules--- such as 16 digit numbers that meet credit card checksum requirements, medical billing codes, etc.

**Database Fingerprinting:** This technique takes either a database dump or live data (via ODBC connection) from a database and only looks for exact matches.

**Exact File Matching:** This technique takes a hash of a file, presumably containing sensitive data, and monitor for any files that match that exact fingerprint.

**Partial Document Matching:** This technique is based on cyclical hashing, where a collection of hashes are taken for a consecutive set of the document fragments, overlapped with a predetermined offset. Upon receiving an outbound traffic, the same algorithm will be applied. Matches on hash values would be counted as containing potential sensitive data.

**Statistical Analysis:** This method uses machine learning techniques, Bayesian analysis, and other statistical methods to analyze a corpus of content and find policy violations based on content that resembles the protected ones.

**Conceptual/Lexicon:** This technique uses a combination of dictionaries, rules, and other analysis to protect "similar" content.

## Policies

As DLP products is policy---based, it must also have a policy---engine to describe what data needs to be classified and what kind of action needs to take upon a policy---violation. All DLP products have a collection of built---in policies, which are further combined into various pre---built categories along with rules, dictionaries so that they can be easily applied to common types of sensitive data with minimal configuration. Usually, DLP product comes with an application that allow a user to configure the policies via a GUI interface. For a few products, e.g., CA DLP, which we can obtain some developer and user documents, the rules is written in a declarative language, such as in XML or proprietary syntax .

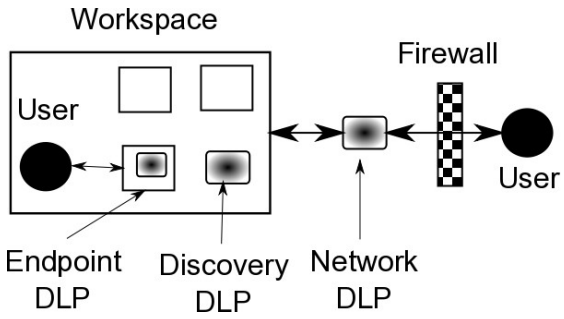


Figure 5: DLP Deployment Architecture

# Background on Secure Medical Workspace

The Secure Medical Workspace (SMW) SRW [1,2,3](figure 6) is a comprehensive solution to the data security challenges that arise in provisioning clinical data for research purposes. The SMW was developed jointly by RENCI and UNC's NC Tracs Institute and motivated by requirements at both UNC and Duke University. In translational and clinical research, patient data is often provisioned to researchers through sanctioned mechanisms, such as on encrypted file systems residing behind institutional firewalls or through secured email. However, once provisioned a significant security risk exists due to the inadvertent or purposeful exposure of data due to lost or stolen portable storage devices (e.g., laptop computers, USB drives) or hard copies of data that had been printed. This security risk can render ineffective methods in place to safeguard data in institutional repositories.

The SMW model for provisioning of research data was developed as a solution to this 'data leakage' problem. In this model, provisioned research data is provided to researchers on virtual machines that that allow for monitoring and controlling the movement of data out of the virtual environment by sophisticated data leakage technologies. The SMW is currently operational at UNC and being used for collaborations with external entities and for internal studies.

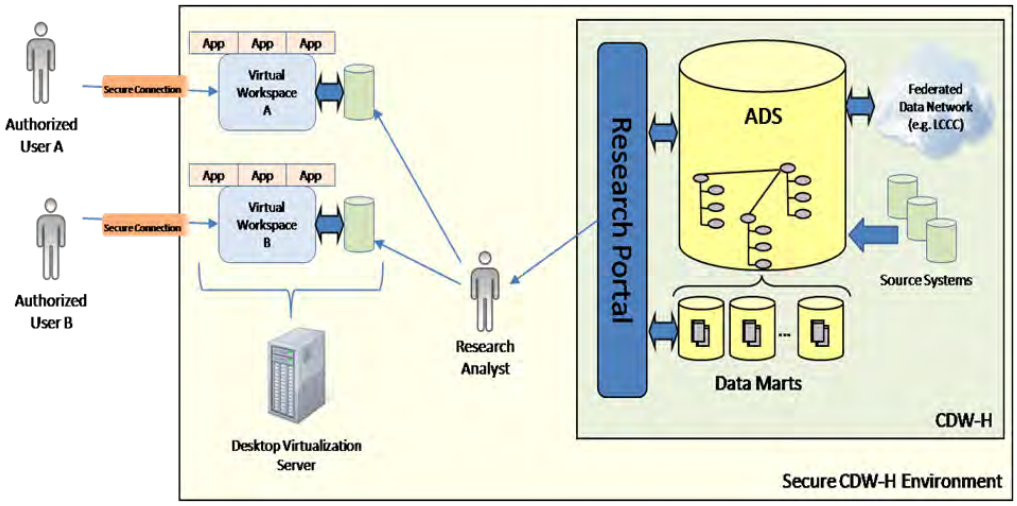


**Figure 6: Secure Medical Workspace --- High Level View**

Significant security features of the SMW include:

- Virtualization technologies to facilitate the set---up, data provisioning, management, and tear--- down of protected virtual workspaces;
- The ability to provide clean virtual desktop images in order to mitigate the possibility that viruses and malware may expose data to the outside world;
- Endpoint Data Loss Prevention (DLP) technologies and techniques to prevent unauthorized use and/or transmission of data, in order to maintain compliance with University policies regarding Sensitive Information;
- The ability to require investigators to acknowledge that any removed data, such as figures and graphs, do not contain sensitive data;
- Standard set of operating systems and tool sets (golden images) that allow support staff to focus on hardening those technologies and to rapidly deploy them;
- Remote access technologies to further isolate the data; all work (e.g., analysis) involving sensitive data is confined to the SMW; the researcher is able to manipulate the data but can remove, copy, or print sensitive data from the SMW only based on institutional policies;
- The ability of compliance officers to review and audit data usage and movements to ensure institutional policies are followed.

A key requirement is that the SMW environment must present a challenge to the user at the point in time that they attempt to remove data from the workspace and the SMW environment must allow the user to override the environment (with justification and auditing) when removal of data is time sensitive and critical to business requirements. These requirements are only met by the Endpoint version of DLP.

## Evaluation Criteria for End Point Data Leakage Protection

The Gartner Group has conducted regular evaluations of DLP technologies for several years. Their criteria for evaluation are listed below and the results of their evaluations are summarized in Figure 3 and 4 --- the Gartner Magic Quadrant for Content Aware DLP vendors from 2013 and 2009 respectively. We have selected those vendors from the 2013 Gartner Group survey that appear in the upper right hand quadrant of figure 3 for further evaluations. The upper right quadrant

represents vendors that have both the ability to produce high quality DLP technology and that are leading the field in terms of new advances in DLP technology. Our rationale for selecting these vendors for further valuation is two--- fold. First, the Gartner Group has found that the companies in the upper right quadrant have stayed in this quadrant over time, with a few new vendors joining the quadrant based on more recent offerings. Second, is that the existing vendor solutions for DLP for the secure workspace are not entirely sufficient to meet our requirements and thus the vendors in the top right quadrant are most likely to come close to meeting our full set of requirements in the future.

## Gartner Groups Criteria

The criteria for a vendor to be included in the Gartner Here Magic Quadrant for content---aware DLP are below and based on features that were generally available as of 29 February 2012:

- Can detect sensitive content in at least two of network traffic, data at rest or endpoint operations
- Can detect sensitive content using at least three of the following content---aware detection techniques, including partial and exact document matching, structured data fingerprinting, statistical analysis, extended regular expression matching, and conceptual and lexicon analysis
- Can support the detection of sensitive data content in structured and unstructured data, using registered or described data definitions
- Can block, at minimum, policy violations that occur via email communication

Vendors must also be determined by Gartner to be significant players in the market, because of market presence or technology innovation. Vendors were excluded from this Magic Quadrant if their offerings:

- Use simple data detection mechanisms (for example, supporting only keyword matching, lexicon, or simple regular expressions)
- Have network---based functions that support fewer than four protocols (for example, email, instant messaging and HTTP)
- Primarily support DLP policy enforcement via content tags assigned to objects

## SW Requirements

The following are requirements of importance for the SMW environment for DLP solutions.

1. Ability to block and/or challenge data loss from an end---point client regardless of how the data is removed from the client. This includes via file copy, applications such as ftp, via user---created software (e.g., that might open a socket for external communications), email, ims, through a web browser web form, or any other means.
2. Support for end---point clients running Windows, Mac OS, or Linux operating systems.
3. Support for end---point clients running in a virtualized environment.
4. Ability to log all attempts to remove data from the end point client. Audit logs should include the time, user, application, policy violations (if one occured), and a snapshot of the data being removed (e.g., first 100 lines for large files).
5. Ability to allow administrators to review policy violations, with tools that facilitate sorting and ranking of violations to enable easy and fast review.
6. Ability to set white lists that allow specific types of data removal from the end---point client to be allowed. Of critical importance is the ability to white lists directories on attached storage

devices, including network attached drives, as places where data can be removed to without blocking/challenge.

7. Ability to prevent viruses and malware that are installed on the end---point client from disabling the DLP protection
8. Ability to prevent standard and administrative user account types from disabling DLP protection.
9. Ability to allow user account types to install software, while preventing users from disabling DLP protection.
10. Ability to create and edit DLP policies
11. Ability to challenge users when a DLP policy infraction occurs. We require that the DLP can display a message to the user when a violation is about to occur (i.e., when the DLP has detected that the user is about to remove sensitive data from the protected end point client). Additionally, the message should be presented as a challenge such that the user can override the DLP and allow the removal of the data to occur. The administrator must be able to set whether a challenge is allowed on a per---policy and per---user/group basis.
12. Ability to customize the user challenge when a DLP policy infraction occurs. The message presented to a user must be modifiable by the system administrator on a per user/group basis.
13. Ability to customize the user response to a challenge. When presented with a challenge, the user should be able to override the challenge and allow the removal of data to occur. In such a case, the user should have to provide a justification for the data removal. The justification should be selected from a list of possible choices that include the ability of the user to provide a free---text response. The administrator should be able to set the list of possible choices.
14. Ability to create custom policies through application programming interfaces. We should be able to provide methods external to the DLP technology for informing the DLP technology as to whether a particular file should be blocked/challenged when an attempt is made to remove the file.

Through our reviews in 2009, 2013, and from Gartner Groups evaluations, we have found that requirements that are satisfied by one vendor are typically satisfied by all vendors, or at least by those in the upper right quadrant of the Gartner Groups' 2013 evaluation. While there are vendor to vendor differences, these differences have been of minor importance for the SMW implementation. For instance, all DLP technology vendors that we have looked at have the ability to let administrators create and modify policy settings and all have fairly intuitive GUI based mechanisms for setting policies.

*Generally, we have seen DLP technologies lacking for the following requirements:*

2. Support for Linux is typically not present and many vendors have stated that they only intent to address DLP for Linux users through network based DLP. Support for Mac OS has been claimed by a couple of vendors, but we have yet to validate such solutions in tests.

7. All DLP technologies that we have seen run within the end---point client and are thus vulnerable to attack from end---point based viruses and malware.

8. Generally this is true for non---administrative accounts on Windows based machines but is not true on administrative accounts. It is unlikely that this can be satisfied for DLP solutions that run within the end point client for Windows based machines. MacOS had not been tested.

9. It is unlikely that this can be satisfied for DLP solutions that run within the end point client for Windows based machines.

12. Not all DLP vendors allow full customization of the message text.

13. Not all DLP vendors allow full customization of the challenge responses.

14. In 2009 we identified no vendors that met this requirement. In 2013, we have seen a few vendors starting to offer solutions based around tagging of files. The introduction of tagging is the most significant advancement for SMW that has occurred between 2009 and 2013 and we discuss this below.

### Vendor provided tagging

While vendor solutions to requirement #14 are still being developed by vendors, it appears that a tagging based approach is increasingly a solution that will be favored by vendors. In this approach, data can be tagged in a way that identifies the data as being of relevance for a defined policy. For example, a policy may state that all files generated by Microsoft Excel should be prevented from being removed from an end---point client using email. The DLP enforcement engine can then check whenever any file is attached to email to see if the file is tagged as generated by Microsoft Excel. There are multiple ways in which tagging can be approached; e.g., a vendor may store the tag on the end---point client or in a database, the tag may be one that only the vendor may set or that other applications can set, the tag may be user/group specific and/or policy specific, and the tag may propagate forward to new data derived from already tagged data.

Several issues arise when looking at tagging that impact whether tagging can be used to address requirement #14:

1. Can third party applications tag data or can only the DLP engine tag data?

2. Is the tag secure against tampering by other applications, including malware/viruses?

3. Does tagging work if the end---point client loses communications with the DLP server?

4. Are there any refresh issues between when the data is tagged and when the policy recognizes the tag?

5. Can third party read/view/edit/delete the tags?

6. Are tags unique?

7. Can tagging propagate to derived data sets?

8. Can data not stored in files be tagged, e.g., data in the system clipboard?

## DLP Vendors

The following lists the DLP vendors that we have looked at. Most of these vendors offer DLP products as an enterprise solution. As a first pass, we have reviewed only documentation made available by each vendor on their web sites. In doing so, we have specifically looked to see if they could not meet any of the requirements that most vendors do meet (#s 1,3,4,5,6,10,11). We have not looked at the depth of features that vendors do provide in meeting those requirements as this would be done only for vendors that are included in our final in---depth evaluations. We have also looked at whether any of the vendors have unique solutions for addressing those requirements typically not

met (#s 2,7,8,9,12,13,14); those vendors that do we have done a second round of review. In this second round, we have done a more thorough examiniation of vendor provided documentation to see if the unique solutions are of relevance in the context of the SMW. Based on the second round of review, we have selected vendors that we felt we should have a person---to---person discussion with their technical team and/or conduct an evaluation of their product. In most cases, Customizability of solutions is typically unknown as vendors typically do not make APIs or SKDs available to non---paying customers. This is despite many attempts by our group to talk to sales representatives and technical staff.

### Code Green Networks

URI: http://www.codegreennetworks.com/

DLP Product Page: http://www.codegreennetworks.com/products/truedlp---overview/

Code Green Network has a DLP offering (network, endpoint, and discovery) that includes a focus on healthcare use cases, including a use case at the Cascade healthcare Community Inc. http://searchhealthit.techtarget.com/tip/Using---data---loss---prevention---software---to---comply---with---new---HIPAA---policies. Gartner Group's evaluation targeted Code Green Networks' solution for small to midsize deployments with low complexity use cases. We have seen no other indications of unique capabilities and thus have not further reviewed Code Green Networks.

### Clearswift

URI: http://www.clearswift.com/

Clearswift markets a data leakage protection capability. However, the where not included in the 2009 or 2013 magic quadrant analysis from the Gartner Group. Gartner previously warned its reader that Clearswift has very minimal market presence in US. As such, we have not further reviewed Clearswift.

### GTB Technologies

URI: http://www.gtbtechnologies.com/

DLP Product Page: http://www.gtbtechnologies.com/en/products/the---gtb---data---loss---platform

GTB Technologies have products of all three DLP channels, however, the three products are not integrated as an enterprise DLP suite. Based on pricing, GTB is one of the more expensive solutions. Gartner Group's assessment has had GTB's solution increase in innovation from 2009 to 2013, and overall Gartner has noted positive feedback from clients despite some shortcoming in features compared to other vendors. Of all DLP vendors, GTB appears to be one of the few company that provides a Software Development Kit (SDK) to facilitate the development of third party product to integrate with their GTP lines of product. In reviewing the SDK documents, the GTB DLP does allow administrators to tailor pop---up messages but does not support customization of challenge responses (requirement 13). GTB does not appear to support tagging or external labeling of data files (requirement 14) , rather they emphasize their 'zero---false---positive' fringer printing technology and claim alternative technologies, e.g. tagging, are imprecise and inaccurate (http://www.prweb.com/releases/2013/3/prweb10492352.htm). Pattern---based inspection with compliance table, lexical analysis and extended regular expression is also supported. See details at http://www.gtbtechnologies.com/en/products/sdk.

### CA

URI: http://www.ca.com

DLP Product Page: http://www.ca.com/us/dlp.aspx

CA has a strong DLP offering but has also been criticized in Gartner's report on documentation and interface issues. CA's end---point DLP does provides APIs for extracting classification information, making classified content consumable for third---party IT and secure software products. CA also allows for customization of pop---up notifications used for challenging users when they attempt to violate DLP policies; both the pop---up title and message body can be customized. Finally, CA includes smart tagging which can be defined in triggers(user policies). A smart tag consists of a tag name and one or more values. The value is typically a pattern used to match particular text. Files containing matched text will be tagged automatically. However, we did not find any approach to tag files manually in the related documentation. Tags for most files are stored in the CMS database, but tags for some types of files, such as MS Office documents, are attached to the files as a property.

### EMC/RSA

URI: http://www.rsa.com/

DLP Product Page: http://www.emc.com/security/rsa---data---loss---prevention.htm

RSA is generally recognized by Gartner Group as one of the leaders in the DLP space. Further, Microsoft has formed a partnership with RSA to incorporate RSA's DLP functionalities into their product, which should further enhance RSA as a market leader. We are note that RSA has demonstrated use in hospital setting (http://www.rsa.com/press_release.aspx?id=10589). However, Gartner has noted that the endpoint agent continues to be basic, and clients reported performance and accuracy issues with using some of the advanced content finger printing capabilities on the endpoint. Based on EMC's documentation (v 9.0 of the DLP product), tagging of data files is not supported. Recent allegations that RSA knowingly weakened its own security standards to support NSA activities lowers the confidence in RSA produced security technologies broadly. Given our focus on endpoint technologies, and continued issues in getting direct contact with RSA technical sales, we have decided not to investigate RSA further.

### General Dynamics Fidelis Security Systems

URI: http://www.fidelissecurity.com/

DLP Product Page: http://www.fidelissecurity.com/network---security---appliance/Fidelis---XPS

Fidelis has traditionally focused on network DLP and has one of the higher cost solutions in the DLP space. We have seen no unique solution for end point DLP and thus did not review Fidelis further.

### McAfee

URI: http://www.mcafee.com

DLP Product Page: http://www.mcafee.com/us/products/data---protection/data---loss---prevention.aspx

McAfee has one of the more interesting end point DLP offerings, as evidenced by its move to one of the premier DLP solutions within the 2013 Gartner Group report.

McAfee has several features which are of note for the SMW environment. First, McAfee is one of

the lower cost solutions and has been used in healthcare settings (http://www.mcafee.com/us/local_content/case_studies/library/cs_ellis_hospital_s.pdf). It's EPO Policy Orchestrator management suite is comprehensive and full featured. Its endpoint DLP product can be deployed in a stand --- alone configuration, unlike several other products we have seen that require contact with the server. Finally, we note that the acquisition of McAfee by Intel is of interest in positioning McAfee to take advantage of hardware and virtualization---based security capabilities.

Of particular note, McAfee's end---point DLP includes capabilities for enforcing policies based upon tagging of data sets. We have investigated this approach for applicability to the SMW solution with a McAfee technical team and will begin an evaluation in 01/2014. Based upon discussions with the McAfee technical team, tagging has the following aspects:

- For each policy that uses tagging, a GUID is generated that is specific to that policy
- For any tagged file, the relevant policy---based GUID is stored as file---based metadata within the Windows File System
- Tagging is based on two approaches:
    - Location based tagging tags files that are located in specific locations
    - Application based tagging tags files that are created or edited by specific applications
- Tags will remain with files as they are moved, copied, or converted into other formats.
- Authorized users can manually add/remove tags from files without using tagging rules. The option can only be accessed from the managed computers.

We are currently investigating whether user---level or kernel---level applications outside of McAfee can create, edit, or delete tags.

### NextSentry

URI: http://www.nextsentry.com/

NextSentry is a provider of DLP technology. We have not reviewed them further as they have not appeared in reports from the Gartner Group nor does review of their web site suggest any unique capabilities.

Palisade System

Company URI: http://www.palisadesystems.com

URI: http://www.proofpoint.com/

Palisade offers a traditional DLP solution that appears geared towards simplicity and general use versus attempting to be a leader of new capabilities. Our assessment from the web site matches that of the Gartner Group. As we've seen no unique capabilities of relevance for SMW, we did not review this further.

### Symantec

URI: http://www.symantec.com

DLP Product Page: http://www.symantec.com/data---loss---prevention

DLP Forum: http://www.symantec.com/connect/security/forums/data---loss---prevention---vontu

Symantec has been the leader in Gartner's Magic Quadrant for many years. A strength of Symantec is that they have a focus on integrating DLP capabilities with new technologies, such as cloud, mobile, and virtualized environments. They also have a sigificant focus on regulatory compliance deployments, and according to Symantec's claims, three out of five top health care providers use their products, although its not clear whether they are claiming usage of their DLP product or other Symantec products. Symantec is also very strong on integrating their DLP technology, including network, discovery, and endpoint, into a comprehensive solution with tools to facilitate understanding who is using data and how and to improve the definition and enforcement of policies.

A significant issue with Symantec is that they have one of the higher price solutions and have been traditionally geared towards large organizations. This has made getting evaluation copies, documentation, and technical support challenging. In our attempts to discover solutions to requirements (#s 2,7,8,9,12,13,14) we have seen no specific capabilities from Symantec (e.g., such as tagging) and thus we have not continued with review of Symantec despite their development of a very capable DLP solution.

### Trend Micro

Company URI: http://us.trendmicro.com/

DLP URL: http://www.trendmicro.com/cloud---content/us/pdfs/business/sb_idlp.pdf

TrendMicro was extensively tested in our initial evaluation in 2011 due to the availability of zero---cost evaluation versions. However, in those tests we found a critical error which TrendMicro was unable to fix despite several weeks of iterations with their development team. The error in question came from a use case whereby a user attempts to overwrite a file on an external drive. If the external drive is monitored by the DLP, a challenge should occur and the user should have the choice of overriding the challenge or canceling the data transfer. If the user selected cancel, there should be no affect on the external file, however, in the case with TrendMicro the external file was corrupted (specifically left with zero bytes). We considered in this round revisiting these issues with TrendMicro, especially as TrendMicro has two documented use cases in hospital environments (Centre Hospitalier Universitaire de Quebec, Luton and Dunstable (L&D) Hospital). However, in 2009 TrendMicro was in the lower quadrant of Gartners report and in 2013 TrendMicro has disappeared from Gartners quadrants entirely.

In addition to seeing now further compelling advances in the TrendMicro DLP online literature, we decided to not review TrendMicro further.

### Verdasys

URI: http://www.verdasys.com/

DLP Product Page: https://www.verdasys.com/solutions/dlp---3.0.html

From 2009 to 2013 Verdasys did move into the Gartner upper quadrant for DLP vendors. Based on literature review, Verdasys solution appears to be comprehensive and well executed. Of note, Verdasys claims to have tagging capabilities and the list of tag---related capabilities looks impressive. Of note, Verdasys claims the ability to automate tagging as well as allow end users to set classification status of data when it is created or changed. Support for inheritence of tags is offered and tracking of tags as data is changed or moved is claimed to be supported. Verdasys literature imply the use of forensic and causal linking approaches, although little detail is provided. Further, Verdasys claims to support several individual (or nested) metadata tags and to support rules which can be based upon multiple tags. Verdasys manages tags through a variety of means, e.g., appending tags to document properties, using Alternate Data Stream (ADS) for Microsoft NTFS, or storing tags by encrypting files and adding tags to the encryption header. Verdasys claims support for Windows and Linux. Given the claims, Verdasys comes closest of current DLPs of offering a solution that fits the SMW needs. The Verdasys system does suffer from some weakness. In particular, the solution appears to require use of the comprehensive DLP solution that includes endpoint, discovery, and networking, which makes the final solution much more complex and expensive than imagined for the SMW. The solution requires use of Windows Server Active Directory even for Linux users, a minor issue in data centers that are heavily Unix based. Finally, the tracking of tags as data is moved and copied appears to be done by client based tools that are hidden from the user, but are still vulnerable to attacks from the user or from other attackers. Pricing has been another concern with Verdasys based on costs in 2011, however, current costs where not found. Verdasys does offer hosted solutions that can be managed by Verdasys (MSIP) or by one's own IT team (IPaaS) starting at $19/month per endpoint and IPaaS at $12/month per endpoint, based on volume, which works out to be on the order of $10---15K per year for 50 endpoints, which is high but reasonable compared to the cost of running other DLP solutions in---house.

### Vericept/TrustWave

URI: https://www.trustwave.com/

Gartner reports had TrustWave in the upper left quadrant in 2009 and removed entirely in 2013. In addition, a review of literature publically available on the TrustWave site did not suggest any unique capabilities or capabilities that match other more known vendors in the space. As such, we did not review TrustWave further.

### Websense

URI: http://www.websense.com

# List of Symbols, Abbreviations, and Acroynms

**AMD** Advanced Micro Devices. i, 13, 16, 25, 41
**API** Application Program Interface. iv, 22, 24, 25, 52
**ASLR** Address Space Layout Randomization. 20

**CPU** Central Processing Unit. 13, 14, 16, 19, 25, 27–29, 34, 35, 38
**CR3** Control Register 3. 12, 19, 20

**DHS** Department of Homeland Security. 38
**DLL** Dynamically Loaded Library. 24, 25
**DLP** Data Loss Prevention. ii, 8, 32, 34–38, 43, 52
**DOMCTL** Domain Control. 17
**DVD** Digital Versatile Disc. 42–44, 46, 47

**EPT** Extended Page Table. 27

**FTP** File Transfer Protocol. 21

**GB** A gigabyte, measure of computer data storage. 21, 34, 41, 42

**hDLP** Hypervisor-based DLP. i, ii, iv, v, 7, 8, 13, 16, 18, 19, 36–39, 41–44, 46–49, 51, 52

**I/O** Input/Output. ii, 8, 9, 21, 27, 36, 37
**IRP** Input/Output Request Packet. 9, 36

**KVM** Kernel-based Virtual Machine. 21–23, 25

**MFT** Master File Table. 8

**MSR** Machine Specific Register. 28
**MTF** Monitor Trap Flag. 16

**NTFS** New Technology File System. 8, 11

**OS** Operating System. 5, 8, 11, 12, 27, 28, 31, 35, 52

**PEB** Process Environment Block. 19, 20
**PII** Personally Identifiable Information. 53 **RENCI** Renaissance Computing Institute. 34 **SCP** Secure Copy. 21
**SID** System Identifiers. 19, 31
**SMW** Secure Medical Workspace. ii, 22, 24, 32, 37
**SSE** Streaming SIMD Extension. 14, 15 **TLB** Translation Lookaside Buffer. 12 **USB** Universal Serial Bus. 28, 29
**VM** Virtual Machine. 8, 9, 12–14, 16, 21, 23, 24, 29, 34, 35, 41, 44
**VT-x** Intel Virtualization Technology for IA-32 and Intel 64 Processors. 41